AFRL-RI-RS-TR-2016-076

CONTEXT-AWARE ACTIVE AUTHENTICATION USING TOUCH GESTURES, TYPING PATTERNS, AND BODY MOVEMENT

LOUISIANA TECH UNIVERSITY

MARCH 2016

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2016-076 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
ANNA L. WEEKS
Work Unit Manager

/ S /
WARREN H. DEBANY, JR.
Technical Advisor, Information
Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MARCH 2016	FINAL TECHNICAL REPORT	SEP 2013 – SEP 2015

4. TITLE AND SUBTITLE

Context-Aware Active Authentication using Touch Gestures, Typing Patterns, and Body Movement

5a. CONTRACT NUMBER
FA8750-13-2-0274

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62722F

6. AUTHOR(S)

Mike O'Neal, Kiran Balagani, Md Karim, Vir Phoha, Andrew Rosenberg, Abdul Serwadda

5d. PROJECT NUMBER
AAP2

5e. TASK NUMBER
LA

5f. WORK UNIT NUMBER
TK

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Louisiana Tech University
305 Wisteria St.
Ruston, LA 71272

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIGB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2016-076

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Design, develop and evaluate a desktop based active authentication system using keystroke timing based biometric features. In addition, various analyses, such as solution scalability and stability, longitudinal degradation, impact of linguistic and cognitive context, feature transformation, and review of population statistics based attacks were performed.

15. SUBJECT TERMS

Active Authentication, behavioral biometrics, keystroke dynamics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**ANNA L. WEEKS**
U	U	U	UU	57	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

We sincerely thank Mr. Richard Guidorizzi, former Program Manager, DARPA, for providing the opportunity and support to conduct research under the auspices of DARPA's Active Authentication program, and Dr. Angelos Keromytis for continuing program management after the departure of Mr. Guidorizzi. We also sincerely thank Ms. Anna Weeks, Air Force Research Laboratory, and Ms. Susan Kloss, DARPA, for their help and support throughout the course of this project.

1 SUMMARY

The goal of this project was to design, develop and evaluate a desktop active authentication system that uses the following keystroke timing based biometric features:

i. Basic keystroke latencies
ii. Cognitive-Pauses in typing, and
iii. Demographic indicators derived from keystroke timings

Using the above listed features we develop a multi-classifier authentication solution. Each classifier in our solution makes a binary decision determining whether a typist is a genuine user or an imposter. Individual classifier decisions are combined using weighted fusion. Decisions are continuously made after a fixed number of keys are typed.

While designing our solution we used a typing dataset collected at Louisiana Tech University in seven separate sessions over a period of three years. Additional tests were performed using a richer dataset collected at the West Point.

In addition to evaluating the authentication performance of our solution, various analyses, such as system consistency, longitudinal degradation, impact of linguistic and cognitive context, feature transformation, and review of population statistics based attacks were performed.

2 INTRODUCTION

2.1 Background and Motivation

Traditional approaches to authentication in which a password, PIN, or fingerprint scan is entered prior to the be-ginning of a user-session cannot prevent unauthorized access if authenticating credentials are stolen or a logged-in computer or device is left unmonitored. A solution to this problem is in-session authentication of users at frequent intervals, a process known as active authentication. Adopting conventional solutions such as password or smartcard based authentication for active authentication is not practical since such solutions would require frequent user interruption. We have developed a production pipeline for actively authenticating users based on their keystrokes as they normally type.

There exist several works on keystroke based authentication in the research realm [1]–[9]. However, results reported in them were derived from small populations; in our investigation, the reported performance of their approaches significantly drop when applied to larger populations. We develop a keystroke based multi-classifier solution for active authentication and achieve very high performance with respect to the key authentication metrics recommended in the literature [10]–[12]. In addition, we show that the performance of our solution remains stable across a variety of distinct populations of various sizes indicating that the proposed solution is scalable.

2.2 Performer Sites, Team and Deliverables

2.2.1 Performer Sites

1. Louisiana Tech University (LTU), Ruston, LA
2. Syracuse University (SU), Syracuse, NY
3. New York Institute of Technology (NYIT), New York, NY
4. City University of New York (CUNY), New York, NY

2.2.2 Team

Investigators

1. Mike O' Neal (PI), Louisiana Tech University
2. Vir V. Phoha (Co-PI), Syracuse University
3. Kiran Balagani (Co-PI), New York Institute of Technology
4. Andrew Rosenberg (Co-PI), City University of New York

Senior Personnel

1. Md Enamul Karim (Researcher), LTU
2. Paolo Gasti (Researcher), NYIT
3. Abdul Serwadda (Post-doc), SU
4. Aaron Elliott (Lead Developer), Aegis Research Labs LLC

Graduate Student Researchers

1. Rajesh Kumar, SU
2. Sujit Poudel, SU
3. Zibo Wang, LTU
4. Diksha Shukla, SU

Undergrad Student Developers

1. Azriel Richardson, LTU
2. John Hawkins, LTU
3. Daniel Adams, LTU
4. Christian Dean, LTU
5. Andrew Duryea, LTU
6. Nick Henry, LTU
7. Sean Manteris, LTU
8. Anna Whitaker, LTU

2.2.3 Deliverables

2.2.3.1 Software, monthly reports

All monthly reports have been uploaded to the TFIMS system. The active authentication pipeline (software) was delivered to Novetta and integrated into their biometric platform.

2.2.3.2 Publications

The list of papers, already published, is provided below:

1. A. Serwadda, V. V. Phoha, S. Poudel, L. Hirshfield, D. Bandara, S. E. Bratt, M. R. Costa, BTAS 2015, fNIRS: A New Modality for Brain Activity-Based Biometric Authentication
2. Brizan, D.G., Goodkind, A., Koch, P., Balagani, K., Phoha, V. V., & Rosenberg, A. (2015). Utilizing linguistically-enhanced keystroke dynamics to predict typist cognition and demographics. International Journal of Human-Computer Studies.
3. Goodkind, A. and Rosenberg, A. (2015). Muddying The Multiword Expression Waters: How Cognitive Demand Affects Multiword Expression Production. Proceedings of NAACL-HLT.

4. Goodkind, A., Brizan D.G. & Rosenberg, A. (2015) Improvements to Keystroke-Based Authentication By Adding Linguistic Context. 7th IEEE Conference on Biometrics: Theory, Applications and Systems (BTAS 2015).
5. Goodkind, A., Brizan D.G. & Rosenberg, A. (under review). Utilizing Overt and Latent Linguistic Structure to Improve Keystroke-Based Authentication. Image and Vision Computing: Best of Biometrics Issue.
6. Locklear, H.; Govindarajan, S.; Sitova, Z.; Goodkind, A.; Brizan, D.G.; Rosenberg, A.; Phoha, V.V.; Gasti, P.; Balagani, K.S., "Continuous authentication with cognition-centric text production and revision features," in Biometrics (IJCB), 2014 IEEE International Joint Conference on, vol., no., pp.1-8, Sept. 29 2014-Oct. 2 2014, doi: 10.1109/BTAS.2014.6996227
7. Jaroslav ˇSedˇenkay, J., Balagani, K. S., Phoha, V., Gasti, P., "Privacy-Preserving Population-Enhanced Biometric Key Generation from Free-Text Keystroke Dynamics," in Biometrics (IJCB), 2014 IEEE International Joint Conference on, 2014

3 DATA COLLECTION

No data was collected at the performer's site during the execution of the project. The experiments were performed using the following two datasets:

(i) Louisiana Tech Dataset
(ii) West Point Dataset

3.1 Louisiana Tech Dataset

Table 1. Louisiana Tech dataset by phase

Time Period	Phase	1st	2nd	3rd	4th R	5th	6th	7th	Total
9/09 -10/09	Phase I	1,000							1,000
4/10 – 5/10	Phase II	500	670						1,170
10/10 – 11/10	Phase III	555	381	254					1,190
4/11 – 5/11	Phase IV	373	450	143	162				1,128
9/11 – 10/11	Phase V	586	223	216	72	134			1,231
4/12 – 5/12	Phase VI	374	386	94	67	59	48		1,028
10/ 2012	Phase VII	310	197	97	57	31	29	31	752
9/09 -10/12	Phase I-Phase VII	3698	2307	804	358	224	77	31	7499

3.2 West Point Dataset

The active authentication pipeline (software) developed for this project was tuned using the Louisiana Tech Dataset which was collected in a single application context. To test the performance of our solution under more realistic scenarios a multi-application context dataset was collected by DARPA from 63 cadets at West Point from January 24, 2015 to March 12, 2015. A keyboard sensor was developed in collaboration with Novetta, and integrated into Novetta's Active Authentication API for keystroke data collection. The equipment was used by the cadets considered of DoD issued laptops. An average of 25.27 +/- 9.44 days of typing data and 168286 +/- 102731 keystroke events per user of the 63 was collected. Application-wise statistics are provided below.

Table 2. Application wise statistics for West Point keystroke dataset

ApplicationName	ApplicationUserCount	AvgUserKeystrokeCount
ONENOTE	2	14105
Borderlands2	2	34895
WINWORD	**62**	**73997**
OUTLOOK	**58**	**24891**
Chrome	28	29460
Spotify	3	7085
Acrobat	13	10050
POWERPNT	10	2636
explorer	17	8442
Firefox	24	25029
iexplore	**53**	**19317**
Mathematica	**39**	**7798**
EXCEL	16	5401

4 BASIC KEYSTROKE LATENCIES

We adopt a multi-classifier based design. Our solution includes five verifiers (i.e., authentication algorithms) that are recommended in the literature for keystroke based authentication [2] [6] [4] and seven types of keystroke timing features. They form thirty five combinations of verifier-feature pairs and produce thirty five decisions.

Our decision to construct a multi-classifier based system is motivated by the findings in [13] that the biometric performance of an individual may significantly vary from use of one algorithm to another algorithm. In addition, numerous studies found that multi-classifier systems, in general, produce better decisions than a single classifier [14]–[18]. Fusion of decisions from multiple classifiers, however, adds new challenges. Specially challenging is accounting for the decision correlations among different classifiers. There are several suggested solutions [15]–[18]. We use a method that naturally normalizes the effect of classifier correlations through the proper weighing of individual classifier decisions during the decision fusion process [18], [19]. This process allows individual classifiers to be more or less contributing than others, and can go so far as to completely ignore particular classifiers (i.e., by assigning a weight=0).

Since exhaustive weighing is an NP-Hard problem [19], weighing the decisions of 35 classifier-feature pairs using a large data set is impractical. We use a hill climbing process for weight approximation and fine-tuned the results with threshold adjustments.

4.1 Feature Sets

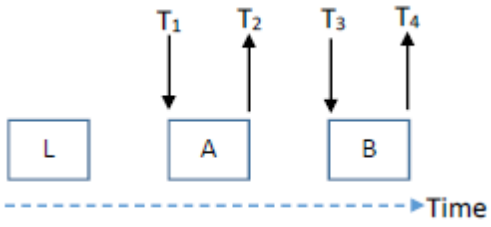


Figure 1. Keystroke latencies

A keystroke timing feature is determined by the press and release times of the associated keys. It can be a unigraph involving just one key, digraph involving two consecutive keys or multi-graph involving more than two consecutive keys. While higher graphs provide more distinguishing contexts, their availability (i.e., frequency of a multi-graph in a given text) is low which makes them less usable. In addition, aggregate digraphs lose information (e.g., it is possible that two timings, $t_1 + t_2 = t'_1 + t'_2$, however, $t_1 \neq t'_1$) and yield lower discriminability. In general, digraphs provide a compromise between context and availability and are popularly used for keystroke based authentication [2], [4], [6]. Although the authors in [20] claim that digraphs do not work well for free texts if their word contexts are not considered, our findings do not support that result. We considered both unigraph and digraph features in our study. We also considered unigraphs with adjacent key context and found them to outperform other keystroke timing features reported in the literature.

Altogether we investigated seven types of timing features as defined below:

i. Key hold latency (KH) – the time elapsed between press and release of a given key,
ii. Inter key latency (IK) – the time elapsed between the release of a key and press of the next key,
iii. Key press latency (KP) – the time elapsed between the presses of two consecutive keys,
iv. Key release latency (KR) – the time elapsed between the releases of two consecutive keys,
v. Key hold latency with next key context (KH_{next}) – KH features for a given key, grouped by the next adjacent character,
vi. Key hold latency with previous key context (KH_{prev}) – KH features for a given key, grouped by the previous adjacent character,
vii. Key hold latency with word context (KH_{wc}) – KH features for a given key, grouped by the word in which the character occurs.

In Figure 1, KH for A is (T_2-T_1), KP for AB is (T_3-T_1), IK for AB is (T_3-T_2), KR for AB is (T_4-T_2), KH_{next} for AB is (T_2-T_1) and KH_{prev} for AB is (T_4-T_3).

4.2 Authentication Algorithms

We studied a set of five authentication algorithms that have been recommended in the literature [9-11] for keystroke based authentication. Two of these algorithms are distance based (SM: Scaled Manhattan [2] and SE: Scaled Euclidean [2]), two of them are matching ratio based (A: Absolute [4] and S: Similarity [6]) and the remaining one is order based (R: Relative [4]).

Scaled Manhattan and Scaled Euclidean, the two distance based algorithms, respectively compute the Manhattan and Euclidean distances between profile feature vector **p** and sample feature vector **s** where, **p** is comprised of the average feature values that are recorded in the user's profile and **s** is comprised of the average feature values computed for the given test sample. To make sure that no feature disproportionately contributes to the measured distance, each feature's contribution is scaled based on its range of values as recorded in the profile.

Absolute and Similarity algorithms make an authentication decision based on the ratio of the number of valid matching pairs to the number of total matching pairs for a given sample.

Each instant of a feature value extracted from a given sample consists of a matching pair with the value of that feature from the user's profile if that user's profile has an entry for that feature. A matching pair is considered to be valid if the corresponding values are "close", where "close" is determined in the "Absolute" algorithm by a predetermined ratio and in the "Similarity" algorithm by a difference computed on the variance of the values of the associated feature as recorded in the profile.

The Relative algorithm creates two ordered sequence of features based on their values in p and s. It then computes the difference between the locations of a feature in those ordered sequences. An authentication decision is made based on the summation of those differences. The idea is that even if someone's typing speed changes, the relative speed for different keys may still be preserved.

4.3 Decision Threshold Determination

Once a verification algorithm computes a score for a keystroke-sample, it has to make a decision whether that sample belongs to an enrolled/genuine user or not. The standard approach is to establish a decision threshold to determine whether a score belongs to the genuine user or not, so that the system objective (e.g., authentication accuracy or EER or HTER, etc.) is optimized. While computing the thresholds we choose to optimize HTER. HTER stands for Half of Total Error Rates and is computed by averaging False Accept Rate (FAR) and False Reject Rate (FRR). HTER approximates ERR (Equal Error Rate) but is computationally less expensive and more practical [21]. We examined one user-specific HTER, one population based HTER, and one hybrid threshold computation method. The hybrid method, namely the K-Chen method, computes user specific thresholds using user-specific statistics but population based parameters.

1) HTER Optimized Thresholds: Population Based and Individual Specific: For each classifier-feature pair we compute a set of scores for the training data set. Then, for computing the user specific threshold for a given classifier-feature pair, we consider the values located between the minimum and maximum scores range for the associated user for that classifier as the candidate

thresholds. Typically, we start with the minimum classifier score and increase its value in small increments. Considering each of those values as thresholds we count the number of imposter and genuine scores that are correctly classified with respect to those thresholds and compute corresponding HTERs. The threshold that generates the lowest HTER is considered to be the final threshold. The population based HTER method computes thresholds in a similar manner, scanning all users' scores available in the training dataset (instead of by looking at just one individual's score) to determine the threshold.

2) K-Chen Thresholds: The K-Chen method [21] is a user-specific threshold computation method that uses user-statistics along with several population based parameters. This method has been successfully used in speaker verification systems. The K-Chen method is inspired by Fururi's threshold which proposed using user-specific imposter mean scores and their standard deviation, and a set of population specific parameters for computing user specific thresholds. The K-Chen method improved Fururi's method incorporating an additional user-specific statistic – mean of genuine scores. These two methods are expressed by the following two equations, respectively:

$$T_{Fururi} = \alpha(\mu' + \sigma') + \beta$$
$$T_{KChen} = b(\mu' + a\sigma') + (1-b)\mu$$

In the above equations, μ, μ' and σ' respectively are the mean of genuine and imposter scores and standard deviation of imposter scores. α, β, *a* and *b* are population specific parameters empirically computed such that the corresponding thresholds produce the best accuracies. Alternatively thee values can be set in such a way that the resultant thresholds match with the HTER thresholds.

Another approach for computing user-specific thresholds discussed in the literature is to normalize user scores with respect to a reference model. However, creating an appropriate reference model is difficult [17] and the resultant thresholds may even degrade performance [18].

4.4 Decision Fusion

Finally, decisions from all verifier-feature pairs must be combined, or fused together, to make an overall authentication decision. We used a weighted decision fusion method that naturally takes care of issues with decision-correlations among various classifiers [18]. We implement weighing of classifier decisions using the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [22]. The advantage of SPSA is that it depends on the measurement of the objective function, not on its gradient and is not affected by the dimension of the gradient vector. However, because it's an approximation algorithm, it is possible that the average HTER achieved using this algorithm may not be optimal. If, however, the optimal HTER is attained, changing the associated fusion threshold will not improve the average HTER further; otherwise it may. Based on this observation, once classifier decision weight assignment is complete, we re-adjusted the fusion threshold if doing so resulted in a lower average HTER .

4.5 Results

4.5.1 Louisiana Tech Data – Single context, Non-Longitudinal Experiment

4.5.1.1 Authentication Performance Criteria

Three primary performance criteria, identified in the literature, for active authentication include: [10]–[12]: (i) time-to-authenticate, (ii) time-to-unauthenticate, and (ii) false accept and reject rates

Since the desired goal of active authentication is to facilitate access control, the user should be continuously authenticated if possible - authenticated from moment to moment. The frequency with which authentication events actually occur in practice determines how closely the system approximates that desired goal. "Time-to-authenticate", which is defined as the period of time between two authentication events, can be used to measure how closely the system approximates continuous authentication. The smaller the time-to-authenticate value the better.

"Time-to-unauthenticate" refers to the amount of time required to identify an imposter once the imposter starts typing. This criteria determines how long an imposter may have access to a system guarded with keystroke based active authentication solution.

A high false reject rate, where the authorized user is incorrectly flagged as a potential imposter, can be a real nuisance because it can lead to frequent user interruptions. A high false accept rate, where an imposter is incorrectly judged to be the authorized user, is even more problematic as it defeats the entire purpose of active authentication.

4.5.1.2 Specific subset of dataset used in this experiment

The data used to train our algorithms comes from the Phase VI and VII datasets of Louisiana Tech and consists of 736 subjects. Data from 488 of the 736 sub-jects was collected from April 18 through May 8th 2012. Data from the remaining 277 subjects was collected between October 15, 2012, and October 31, 2012. The April/May and October groups are completely distinct in that no individual subject is counted as participating in both phases.

Each of the 736 subjects included in this study typed in two separate sessions: the gap between the two sessions ranged from 0 to 19 days. In each session the participants typed at least 300 characters in response to each of twelve questions. Thus the minimum number of keystrokes collected per subject per session was 3,600. However, most of the subjects felt compelled to complete their answers to the questions and over-typed. The average number of keystrokes per user for each of the different sessions is presented in Table 3.

Table 3. Average keystrokes per user

April-May 2012 Phase 1 Session 1	April-May 2012 Phase 1 Session 2	October 2012 Phase 2 Session 1	October 2012 Phase 2 Session 2
5,222	5,085	4,820	4,704

4.5.1.3 Authentication Performance

Authentication performance metrics for three different threshold schemes are presented below. These results are rigorously verified and independently replicated. Use specific HTER performs the best with average HTER <0.011.

Table 4. Authentication performance metrics

Threshold Method	FAR	FRR	HTER
User Specific HTER	0.007921923	0.013292899	0.010607411
K-Chen	0.013946761	0.015217012	0.014581887
Population-based HTER	0.027285169	0.01897904	0.023132105

4.5.1.4 Distribution of HTER

Multiple recent studies [24] [25] demonstrate that the average performance of a given biometric system is disproportionately affected by a small segment of users. We find the same to be true for keystroke base biometrics. Figure 2 shows that for our individual specific HTER threshold, 21.74% of the participants generated no error, 66.4% of the participants performed better than the population average (HTER .01); a small percentage of the participants performed less well increasing the overall average error rate.

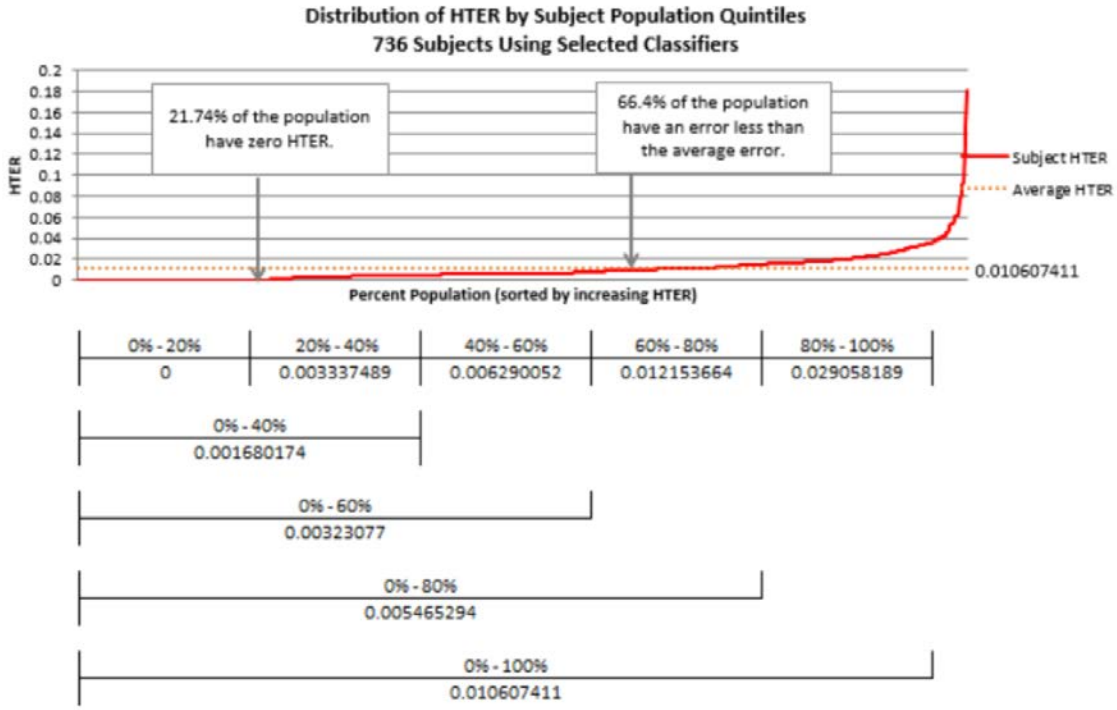


Figure 2. Distribution of HTERs

A closer look to the worst quintile of the performers in Figure 3 reveals that 99.6% of the

population have an overall HTER of less than 0.1. In fact, only three out of 736 users had an HTER greater than 0.1 and only one had HTER greater than 0.15.

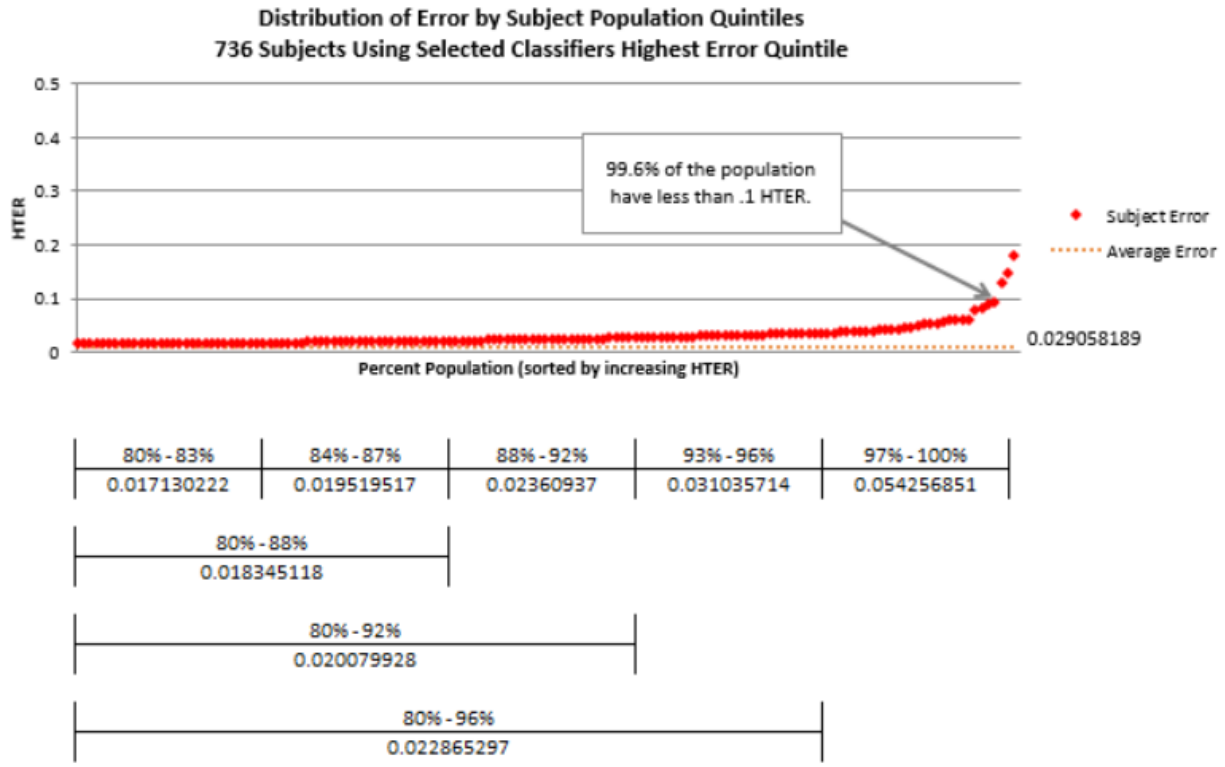


Figure 3. Distribution of HTERs for the worst 20% performers

4.5.1.5 Time to Authenticate

We observe that authentication performance of the developed system nearly saturates around 550 keystrokes. However, it takes on average 200 sec for an average typist to type that many keys. To avoid this delay without compromising accuracy, we use a sliding window of 550 keystrokes and during each authentication decision move the window by 1/10th of its size, i.e., we discard 55 keys from the window and add 55 keys to it in a FIFO manner (Figure 4). Thus after the first 200 sec, time-to-authenticate remains about 20 sec while a user is typing and authentication is always made with 550 keystrokes.

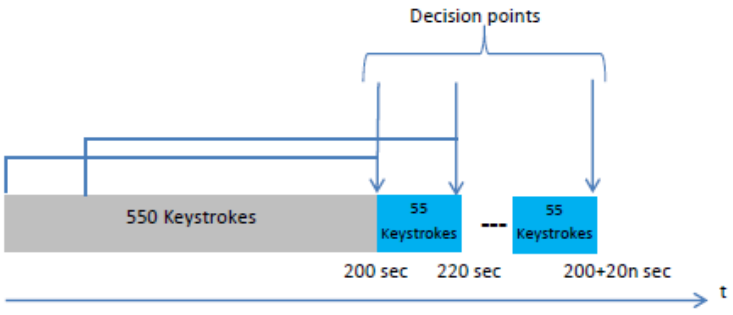


Figure 4. Sliding sample window

4.5.1.6 Time-to-unauthenticate

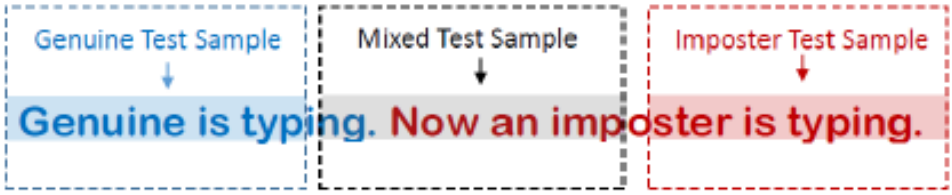


Figure 5. Transition between a genuine and imposter

92.25% of the imposters were identified within first 7 decisions i.e., 385 keystrokes. This means most of the imposters were identified in the mixed test sample region in Figure 5, assuming that the test sample window was filled with genuine keystrokes when the imposter began typing. A histogram showing the distribution of average decisions required to unauthenticate an individual is presented in Figure 6.

Figure 6. Histogram for average number of decisions required to unauthenticate

4.5.1.7 Scalability and Stability

We tested the scalability and stability of keystroke based biometrics using a population based HTER method since we had sufficient data for these tests with this method. We investigated the following:

(i) How overall error rates are impacted if thresholds are computed using populations of different sizes (Table 5)
(ii) How overall error rates are impacted if thresholds are computed using different ratios of imposter and genuine samples (Table 6)
(iii) How overall error rates are impacted if tested on populations of different sizes and non-overlapping populations of identical size (Table 7)

Corresponding results, that are all quite consistent, are presented in Tables 5 through 7.

Table 5. System performance for thresholds computed using different population size

Threshold Population Size	FAR	FRR	HTER
100	0.02717741	0.01967629	0.02342685
300	0.027290005	0.01902619	0.023158098
736	0.027285169	0.01897904	0.023132105

Table 6. System performance for different ratios of training samples

# of samples used for threshold computation		FPR	FNR	HTER
Genuine	Impostor			
40,000	40,000	0.027311095	0.01900014	0.023155617
40,000	30,000	0.027411321	0.01879715	0.023104236
30,000	40,000	0.027255437	0.01901904	0.023137239

Table 7: System performance for various subsets of population

Non-overlapping # of Participants	HTER	Cumulative # of Participants	Cumulative HTER
100	0.010700883	100	0.010700883
100	0.010413961	200	0.010557422
100	0.010511376	300	0.010542073
100	0.010844217	400	0.010617609
100	0.010485292	500	0.010591146
100	0.010700003	600	0.010609289
100	0.01053229	700	0.010598289
36	0.010784816	736	0.010607412

4.5.2 Louisiana Tech Data – Longitudinal Experiment

Gaps of 0 to 19 days between training and testing do not show any significant impact on authentication accuracy. However, study between phase V, VI and VII dataset (data were collected with about six month's separation) shows that, with time, while the FN (False Negative rate) deteriorated, the FP (False Positive rate) either remained almost flat or improved. A closer inspection shows that while some genuine users typing characteristics change over time, the imposter characteristics for a given session remain rather robust across sessions.

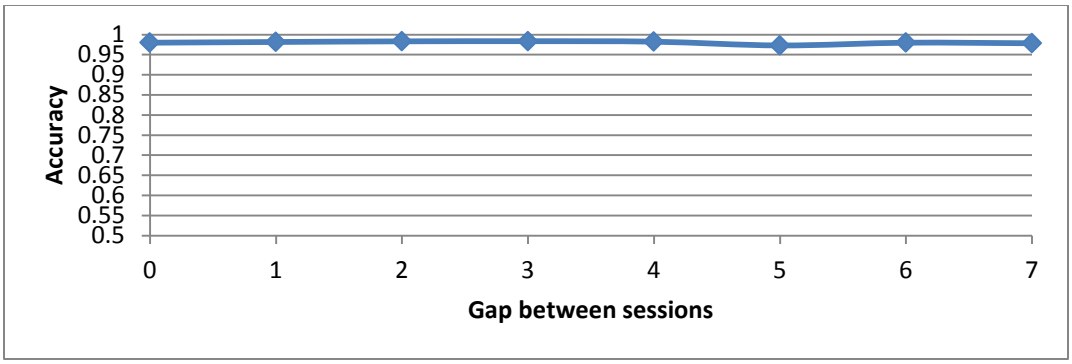


Figure 7. Performance does not degrade for short gaps

Table 8. Results for longitudinal test across sessions (long gaps)

Test Phase	Profile Phase	Number of Users	HTER= (FN+FP)/2	FN	FP	Accuracy
5	5	512	0.039076537	0.049040993	0.029112082	0.960923463
6	5	422	0.054942963	0.078897652	0.030988275	0.945057037
7	5	244	0.070209981	0.116001002	0.02441896	0.929790019
6	6	269	0.051797381	0.011602696	0.091992066	0.948202619
7	6	269	0.049064038	0.018859178	0.079268897	0.950935962

4.5.3 West Point Data – Multiple Context Experiment

4.5.3.1 Applications Context

Our analysis shows that application specific-templates perform better than application-agnostic templates. Our results are presented in the Table 9 below.

Table 9. Authentication performance under different application contexts

AnalysisSetDescription	UserCount	MeanHTER	StDevHTER
West Point WINWORD Session 1, WINWORD Session 2 Typing.	**62**	**0.0582**	**0.0442**
West Point WINWORD Session 1, OUTLOOK Session 2 Typing.	58	0.0810	0.0707
West Point WINWORD Session 1, IEXPLORE Session 2 Typing.	51	0.2766	0.1177
West Point WINWORD Session 1, MATHEMATICA Session 2 Typing.	35	0.3247	0.1451
West Point OUTLOOK Session 1, WINWORD Session 2 Typing.	56	0.0726	0.0584
West Point OUTLOOK Session 1, OUTLOOK Session 2 Typing.	**56**	**0.0589**	**0.0523**
West Point OUTLOOK Session 1, IEXPLORE Session 2 Typing.	47	0.2645	0.1053
West Point OUTLOOK Session 1, MATHEMATICA Session 2 Typing.	34	0.3499	0.1615
West Point IEXPLORE Session 1, WINWORD Session 2 Typing.	51	0.1167	0.0753
West Point IEXPLORE Session 1, OUTLOOK Session 2 Typing.	49	0.1190	0.0932
West Point IEXPLORE Session 1, IEXPLORE Session 2 Typing.	**51**	**0.1375**	**0.0930**
West Point IEXPLORE Session 1, MATHEMATICA Session 2 Typing.	32	0.3015	0.1306
West Point MATHEMATICA Session 1, WINWORD Session 2 Typing.	34	0.2434	0.1532
West Point MATHEMATICA Session 1, OUTLOOK Session 2 Typing.	34	0.2639	0.1581
West Point MATHEMATICA Session 1, IEXPLORE Session 2 Typing.	31	0.3014	0.1410
West Point MATHEMATICA Session 1, MATHEMATICA Session 2 Typing.	**34**	**0.1770**	**0.1104**

In Table 9, we highlight authentication performance under the same authentication context (i.e., the profile and the probe are from the same application) versus authentication performance under different application contexts (e.g., the profile is created from OUTLOOK and probe from Internet Explorer). In all cases, when the application context between the profile and the probe are different, we noticed significant increase in the HTERs, indicating that application context plays an important role in authentication performance. In comparison, authentication agnostic template produced HTER of 0.1468.

Our analysis also shows that training sample size impacts the authentication performance, see HTERs in Table 10 below.

Table 10. Performance of application-specific keystroke authentication

AnalysisSetDescription	UserCount	MeanHTER	StDevHTER
WPS1 WINWORD 800kses, Training Test.	62	0.0548	0.0425
WPS1 WINWORD 1100kses, Training Test.	62	0.0582	0.0442
WPS1 WINWORD 1400kses, Training Test.	62	0.0355	0.0371
WPS1 OUTLOOK 800kses, Training Test.	57	0.0542	0.0462
WPS1 OUTLOOK 1100kses, Training Test.	56	0.0589	0.0523
WPS1 OUTLOOK 1400kses, Training Test.	56	0.0248	0.0316
WPS1 IEXPLORE 800kses, Training Test.	51	0.1051	0.0696
WPS1 IEXPLORE 1100kses, Training Test.	51	0.1375	0.0930
WPS1 IEXPLORE 1400kses, Training Test.	48	0.0514	0.0473
WPS1 MATHEMATICA 800kses, Training Test.	34	0.1638	0.0781
WPS1 MATHEMATICA 1100kses, Training Test.	34	0.1770	0.1104
WPS1 MATHEMATICA 1400kses, Training Test.	26	0.1058	0.0967

In Table 10, we show how the authentication performance is impacted when the training sample size is varied between 800 keystrokes and 1400 keystrokes.

Table 11. Predicting application context using keystroke information (top 50 features).

Including	RF Accuracy	NB Accuracy	KNN Accuracy	TAN Accuracy	RF Acc Cross	NB Acc Cross	KNN Acc Cross	TAN Acc Cross
All	91.75%	88.35%	58.25%	84.47%	95.15%	86.89%	60.68%	92.23%
KH, KPL, KRL	92.23%	85.92%	57.28%	86.41%	95.63%	85.92%	43.69%	93.69%
All but KH	89.32%	83.50%	53.40%	84.95%	92.72%	83.01%	51.46%	90.78%
All but IK	92.23%	87.86%	59.22%	84.47%	95.63%	87.38%	57.28%	92.72%
All but KH2	91.75%	88.35%	60.19%	87.38%	95.63%	85.92%	62.62%	94.17%
All but KPL	93.20%	88.83%	61.17%	84.95%	95.63%	86.41%	56.31%	91.75%
All but KRL	91.75%	88.35%	58.25%	84.95%	95.15%	86.41%	57.77%	92.72%
All but KeyHoldWithPrevVKCode	92.72%	87.38%	62.14%	84.95%	95.15%	85.44%	62.62%	91.75%
All but KeyHoldWithNextVKCode	91.75%	88.35%	61.65%	85.92%	95.63%	87.38%	59.22%	92.23%
STDDEV, NORMALIZEDCOUNT	91.75%	88.35%	63.11%	84.95%	95.15%	87.86%	70.87%	91.75%
MEAN, NORMALIZEDCOUNT	92.72%	89.32%	68.45%	87.38%	95.15%	88.35%	66.50%	92.23%
MEAN, STDDEV	61.17%	57.28%	31.55%	53.88%	63.59%	53.88%	31.55%	63.11%
NORMALIZEDCOUNT	92.23%	89.32%	71.36%	87.86%	95.15%	88.83%	73.30%	92.23%
KH; NORMALIZEDCOUNT	89.81%	88.83%	32.04%	89.81%	92.72%	87.38%	32.04%	90.29%
IK; NORMALIZEDCOUNT	90.29%	89.81%	72.33%	84.95%	89.32%	84.95%	76.70%	86.41%
KH2; NORMALIZEDCOUNT	51.94%	70.39%	64.56%	70.87%	73.79%	66.50%	55.34%	68.93%
KPL; NORMALIZEDCOUNT	88.83%	85.92%	71.36%	80.58%	89.32%	84.95%	77.18%	87.38%
KRL; NORMALIZEDCOUNT	88.83%	86.41%	73.79%	83.98%	88.35%	87.38%	75.73%	87.38%
KeyHoldWithNextVKCode ; NORMALIZEDCOUNT	89.81%	84.47%	69.42%	87.38%	91.26%	86.89%	72.82%	84.95%
KeyHoldWithPrevVKCode; NORMALIZEDCOUNT	85.44%	83.98%	34.95%	81.55%	87.86%	82.04%	48.54%	85.44%
KH; MEAN, STDDEV	54.85%	48.06%	25.24%	48.06%	57.28%	46.60%	24.27%	51.46%
IK; MEAN, STDDEV	51.46%	56.80%	35.44%	52.43%	52.43%	56.80%	30.58%	54.37%
KH2; MEAN, STDDEV	43.20%	44.66%	28.64%	46.12%	46.60%	42.23%	30.58%	43.69%
KPL; MEAN, STDDEV	55.83%	52.91%	31.55%	49.51%	55.83%	53.40%	31.07%	52.91%
KRL; MEAN, STDDEV	59.71%	54.37%	33.98%	50.97%	51.94%	54.37%	32.04%	47.57%
KeyHoldWithNextVKCode ; MEAN, STDDEV	48.06%	48.06%	30.10%	42.72%	51.94%	43.69%	31.55%	52.43%
KeyHoldWithPrevVKCode; MEAN, STDDEV	55.34%	52.43%	33.50%	45.63%	53.40%	43.69%	33.50%	47.57%

Table 12. Predicting application context using keystroke information (top 10 features)

Including	RF Accuracy	NB Accuracy	KNN Accuracy	TAN Accuracy	RF Accuracy Cross	NB Accuracy Cross	KNN Accuracy Cross	TAN Accuracy Cross
All	80.58%	79.13%	61.17%	78.16%	89.81%	80.10%	0.6068	83.01%
KH, KPL, KRL	79.13%	78.64%	63.11%	78.16%	85.92%	81.07%	0.63592	85.44%
All but KH	76.21%	78.16%	63.11%	76.21%	88.83%	81.55%	0.6068	83.01%
All but IK	79.13%	80.58%	60.19%	75.24%	88.35%	80.10%	0.57767	82.52%
All but KH2	**80.10%**	78.64%	61.65%	78.16%	90.29%	82.52%	0.6165	83.50%
All but KPL	**80.10%**	81.07%	60.68%	75.73%	89.81%	80.58%	0.58738	83.98%
All but KRL	79.13%	81.07%	61.65%	77.18%	88.35%	78.16%	0.58738	83.50%
All but KeyHoldWithPrevVKCode	80.10%	78.16%	63.11%	78.16%	87.38%	79.61%	0.60194	81.55%
All but KeyHoldWithNextVKCode	81.07%	79.61%	60.19%	77.67%	90.29%	81.07%	0.59709	82.04%
STDDEV, NORMALIZEDCOUNT	79.61%	**81.55%**	69.42%	78.16%	88.83%	82.52%	0.62621	83.01%
MEAN, NORMALIZEDCOUNT	80.58%	79.61%	63.59%	77.67%	88.35%	80.58%	0.66505	83.01%
MEAN, STDDEV	47.57%	49.51%	31.07%	46.60%	50.49%	44.66%	0.34466	44.17%
NORMALIZEDCOUNT	80.10%	**81.55%**	70.87%	77.67%	86.89%	83.50%	0.71359	83.01%
KH; NORMALIZEDCOUNT	74.76%	69.42%	70.39%	65.05%	76.70%	75.24%	0.68932	67.48%
IK; NORMALIZEDCOUNT	68.93%	74.76%	70.39%	71.84%	74.27%	76.21%	0.71359	70.87%
KH2; NORMALIZEDCOUNT	48.54%	49.51%	59.71%	53.88%	56.31%	50.49%	0.53398	50.00%
KPL; NORMALIZEDCOUNT	68.93%	73.79%	70.39%	68.93%	76.21%	78.64%	0.71359	76.21%
KRL; NORMALIZEDCOUNT	67.48%	73.30%	68.45%	70.87%	78.64%	80.10%	0.71359	69.90%
KeyHoldWithNextVKCode; NORMALIZEDCOUNT	64.08%	70.87%	71.36%	64.08%	67.48%	76.70%	0.70874	65.53%
KeyHoldWithPrevVKCode; NORMALIZEDCOUNT	59.71%	53.40%	57.77%	50.00%	73.79%	66.02%	0.63107	62.62%
KH; MEAN, STDDEV	32.52%	36.89%	32.52%	30.10%	26.21%	30.58%	0.29126	30.10%
IK; MEAN, STDDEV	47.09%	48.54%	33.98%	44.17%	48.06%	47.57%	0.34951	43.20%
KH2; MEAN, STDDEV	36.41%	37.38%	31.55%	38.35%	42.72%	32.04%	0.35437	35.44%
KPL; MEAN, STDDEV	50.49%	47.09%	36.41%	43.69%	50.49%	52.43%	0.36893	42.23%
KRL; MEAN, STDDEV	41.26%	45.63%	34.95%	40.78%	49.51%	50.00%	0.33981	39.81%
KeyHoldWithNextVKCode; MEAN, STDDEV	35.44%	35.92%	28.16%	38.83%	37.38%	37.38%	0.32039	37.38%
KeyHoldWithPrevVKCode; MEAN, STDDEV	34.47%	39.32%	33.98%	30.10%	31.55%	32.04%	0.32039	30.10%

Table 13. Predicting application context using keystroke information (top 100 features)

Including	RF Accuracy	NB Accuracy	KNN Accuracy	TAN Accuracy	RF Accuracy Cross	NB Accuracy Cross	KNN Accuracy Cross	TAN Accuracy Cross
All	94.66%	87.86%	50.49%	85.92%	96.60%	86.41%	0.57767	92.23%
KH, KPL, KRL	94.66%	85.44%	48.06%	86.89%	97.57%	83.50%	0.42233	95.15%
All but KH	93.20%	88.35%	45.15%	85.92%	94.66%	88.35%	0.43204	91.75%
All but IK	94.66%	86.41%	47.57%	88.35%	97.09%	84.95%	0.57282	92.72%
All but KH2	94.66%	87.86%	54.37%	84.95%	96.60%	86.89%	0.64563	94.17%
All but KPL	94.17%	85.92%	48.54%	86.89%	96.60%	85.44%	0.60194	93.20%
All but KRL	93.69%	86.41%	50.00%	87.38%	97.09%	85.44%	0.59223	93.69%
All but KeyHoldWithPrevVKCode	94.66%	86.41%	56.80%	85.92%	95.63%	86.41%	0.54854	91.75%
All but KeyHoldWithNextVKCode	94.66%	85.44%	48.54%	87.38%	97.09%	86.41%	0.56796	93.69%
STDDEV, NORMALIZEDCOUNT	95.15%	88.83%	54.85%	86.89%	96.60%	87.86%	0.64563	92.23%
MEAN, NORMALIZEDCOUNT	94.17%	88.35%	61.17%	87.38%	96.60%	87.86%	0.66019	93.69%
MEAN, STDDEV	58.25%	62.14%	32.04%	57.28%	59.71%	56.80%	0.33495	62.14%
NORMALIZEDCOUNT	94.17%	89.81%	65.53%	87.38%	96.60%	89.81%	0.73301	94.17%
KH; NORMALIZEDCOUNT	90.29%	86.89%	25.73%	89.32%	91.75%	82.04%	0.25243	92.23%
IK; NORMALIZEDCOUNT	93.69%	91.26%	70.87%	91.75%	93.20%	91.75%	0.74272	90.29%
KH2; NORMALIZEDCOUNT	65.53%	71.84%	55.83%	76.21%	78.64%	69.90%	0.49515	76.21%
KPL; NORMALIZEDCOUNT	92.23%	89.32%	70.39%	87.86%	93.69%	92.23%	0.75243	91.75%
KRL; NORMALIZEDCOUNT	93.20%	89.32%	71.36%	87.86%	92.23%	92.72%	0.75243	94.66%
KeyHoldWithNextVKCode; NORMALIZEDCOUNT	93.69%	91.26%	72.33%	90.29%	91.75%	90.78%	0.73786	90.29%
KeyHoldWithPrevVKCode; NORMALIZEDCOUNT	88.35%	84.95%	34.95%	84.95%	90.29%	84.95%	0.47087	90.29%
KH; MEAN, STDDEV	52.43%	50.49%	25.24%	46.60%	55.83%	47.57%	0.24272	50.00%
IK; MEAN, STDDEV	56.31%	56.80%	35.44%	52.91%	55.34%	61.17%	0.30583	52.43%
KH2; MEAN, STDDEV	47.57%	46.12%	31.07%	50.49%	49.03%	44.17%	0.30097	50.00%
KPL; MEAN, STDDEV	60.68%	57.77%	31.55%	58.25%	55.83%	56.31%	0.30583	52.43%
KRL; MEAN, STDDEV	57.28%	58.74%	30.58%	54.37%	53.40%	57.77%	0.31068	52.43%
KeyHoldWithNextVKCode; MEAN, STDDEV	49.51%	49.51%	28.64%	43.69%	53.88%	45.63%	0.29126	51.46%
KeyHoldWithPrevVKCode; MEAN, STDDEV	58.25%	50.49%	27.67%	49.51%	59.22%	42.72%	0.30583	50.00%

Tables 11-13 show the accuracies with which applications can be predicted based on keystroke information. Our results show that *with as few as ten keystroke features*, we were able to predict the application context with approximately 81 percent accuracy, thus highlighting the fact that keystroke features do leak application context information.

5 COGNITIVE PAUSE (PAUSALITY) FEATURES

We define the habitual pauses users' exhibit during typing as the cognitive pause or pausality features. We extract 123 types of pausality features and compute Equal Error Rates varying the length of the pause and at different authentication scan lengths using a dataset consisting of 486 users.

Table 14. EER for pausality features

PP Burst (123 Features)	Authentication Scan Length				
	30 sec	60 sec	90 sec	120 sec	150 sec
PP-200	0.265	0.246	0.144	0.091	0.056
PP-400	0.265	0.219	0.149	0.105	0.070
PP-600	0.267	0.216	0.168	0.118	0.088
PP-800	0.264	0.218	0.180	0.133	0.098
PP-1000	0.258	0.223	0.186	0.150	0.118
PP-1200	0.253	0.218	0.195	0.161	0.126
PP-1500	0.260	0.222	0.195	0.168	0.138

6 DEMOGRAPHIC, LINGUISTIC AND COGNITIVE CONTEXTS

We incorporated linguistic elements (stylometry) into a keystroke authentication system (keystroke dynamics) to refine the performance of basic or low-level keystroke measures. We also investigated how people's typing varies, and what this variation reveals about their cognitive processes and personal attributes (i.e. demographics). Overall, our experiments demonstrated the relevance of linguistic context in classifying keystroke patterns, whether for user authentication or identifying the characteristics of a population. By situating an n-graph within the context of the linguistic structure it was produced within, the set of observations that make up each feature will be more meaningful. We have established a strong baseline for the necessity of taking these factors into account when analyzing keystroke dynamics.

6.1 Methods, Assumptions and Procedures

Keystroke Dynamics Features - Keystroke dynamics looks at the speed at which a user's hands move across a keyboard Bergadano, 2002 and the timing between keystrokes. The features analyzed in the present study include, for example, overall user typing speed, durations and frequencies of pauses in typing, and pauses before specific keys.

Stylometric Features - Stylometry incorporates syntactic, lexical and semantic analyses of a given text. Every aspect from average sentence length to part-of-speech frequency falls under the purview of stylometric analysis. Stylometry analyzes text per se rather than the features of text production.

Production Features - Production features are a hybrid of the above two categories incorporating elements from both linguistic analyses and keystroke rate and timing. While stylometric and keystroke dynamic features are measured independently of one another, production features use elements of both to create unique categories of features. For example,

while stylometric features may look only at the frequency of verbs to nouns, and keystroke dynamics may look only at average keystroke typing speed, production features may measure the average typing speed of verbs versus nouns. The success of this hybridization has been shown in previous studies, Vizer et al., 2009, who used combined features to predict typists' stress levels.

6.1.1 Recognition of Demography

For demographic recognition, we divide our subjects along two broad demographic dimensions: gender (males vs. females) and primary language (native English speakers vs. non-native speakers of English). Each of these demographic divisions may be viewed as a cohort with a different set of keystroke dynamics when compared to the opposite cohort. We aim to be able to place a user in a cohort based on the user's typing patterns and language use. In the context of user identification and verification, this can be used as a filter to eliminate some candidates from further consideration enabling more focused downstream analysis. We use a wide array of features based on keystroke dynamics, stylometry and their intersection.

6.1.1.1 Results

Mean pause time prior to a keystroke is heavily dependent upon the frequency with which a key is used.

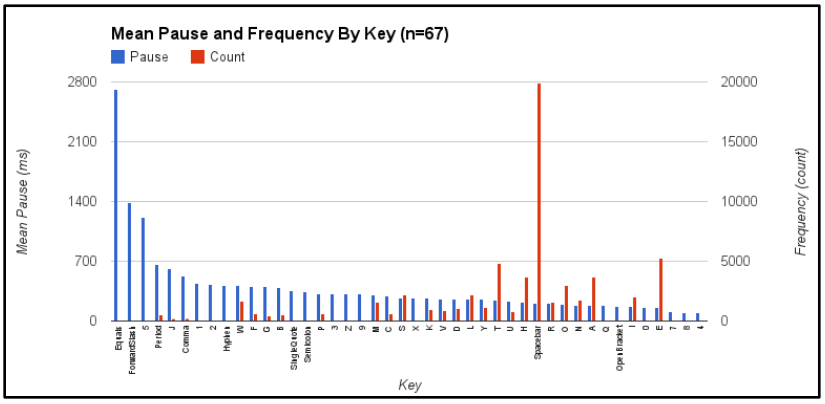


Figure 8. Mean pause and frequency by key

Using this information modest improvements were seen in demographic experiments, but a robust methodology for incorporating predictability was never found.

Experiment Improvements (accuracy %)

1. Handedness: 89.42 → 89.72
2. Gender: 62.16 → 66.03
3. Native Language: 85.1 → 85.4

We utilized a number of different machine learning classification algorithms to increase the accuracy of predicting three demographic markers of our subjects below. The accuracy percentage is the F_1 statistic for the minority class.

1. Gender - Utilizing a Logistic Regression classifier, we improved accuracy from a baseline of 45% to 52% for identifying female subjects.

2. Handedness - Utilizing a Naive Bayes classifier, we increased accuracy from 10% to 22% for identifying left-handed subjects.
3. Native English Language Speakers (Native [L1 English] vs Non-native [L2 English]) - Utilizing a Naive Bayes classifier we improved accuracy from a baseline of 17% to 46% for L2 English speakers.

For 95% of the subjects, we were able to predict at least two of the three demographics correctly.

6.1.2 Recognition of Cognitive Load

In the case of predicting cognitive load, we aim to determine whether the user is performing a rote task, such as recalling known information, or performing a task which is more imaginative and therefore cognitively taxing or something in between. Following Vizer et al. 2009, we expect to see that a user's typing patterns will vary based on the cognitive load of the user. To perform these experiments we ask subjects to perform a variety of tasks with differing expected cognitive demands. We later predict this load for a given typing session.

6.1.2.1 Results

These experiments use the same features as described for demographic recognition.

For this set of experiments, we tried to predict the cognitive complexity of the essay prompt that a subject was answering. Our hypothesis was that the varying cognitive demands would have a noticeable and predictable effect on the timing of keystroke language production. While the tasks were assigned labels of 1-6, we did not approach this as a regression problem, where there was a continuous relationship between complexity levels. Rather, we treated each task as a discrete type, with a general trend from easier tasks to more difficult tasks.

Although each writing prompt was given one of six task labels, we felt that this level of granularity was too fine-grained. Despite this, we were able to predict the specific level of cognitive complexity with 33% accuracy, from an at-chance baseline of 17%. When predicting whether the writing prompt was either the simplest type of task or most complex type, we were able to achieve an accuracy of 72%, from a 50% baseline. Utilizing the numerical label assigned to each task, we were able to achieve an 81% accuracy within a margin of error of 2 levels of complexity.

7 INCLUSION OF LINGUISTIC CONTEXT

Our experiments demonstrated the relevance of linguistic context in classifying keystroke patterns, whether for user authentication or identifying the characteristics of a population. By situating an n-graph within the context of the linguistic structure it was produced within, the set of observations that make up each feature will be more meaningful.

The ideal keystroke atomic unit is digraph hold, encompassing cumulative hold times of two keystrokes in a digraph.

Table 15. EER for cumulative hold

Cumulative Metrics	
Feature Set	EER
Unigraph KeyHold	0.1047
Unigraph Preceding Pause	0.1767
Unigraph KeyHold + Pause	0.1647
Digraph KeyHold	0.0877
Digraph Preceding Pauses	0.1434
Digraph KeyHold + Pause	0.2155
Trigraph KeyHold	0.1337
Trigraph Preceding Pauses	0.2169
Trigraph KeyHold + Pause	0.2398

We also find that fusing multiple atomic-type features improves results. Example raw features are described below based on a sequence of characters, "TH"

1. raw unigraph hold HOLD__H
2. unigraph hold in digraph context HOLD__H_T
3. preceding pause PAUSE__H
4. digraph interval INTERVAL__T_H

Table 16. EER for keystroke latency with context

Feature Set	EER
unigraph hold, intervals	4.8%
unigraph hold in digraph context, interval	4.2%
unigraph - raw & in digraph context, interval	3.8%
all raw features above	**3.6%**

Moreover we find that adding linguistic context to raw features further improves results. For example consider the 'NT' characters in the word WANTED

1. ngraph in word - NT_WANTED
2. ngraph in lemma the unmodified base form of "WANTED" - NT_WANT
3. ngraph in Part of Speech (POS) - NT_PAST_TENSE_VERB
4. ngraph in Pennbaker context - NT_CONTENT

Words are categorized as content (nouns, verbs, etc.) or function words (pronouns, determiners,etc.) Function words are further subdivided by whether they are part of a highly frequent word list, which is hypothesized by James Pennebaker to be highly psychologically informative

Table 17. Various word contexts vs EERs

Feature Set	EER
Raw Holds/Intervals	3.6%
Raw + Word Context	3.2%
Raw + Lemma	3.0%
Raw + POS	4.3%
Raw + Pennebaker	3.6%
Raw + Lemma + Pennebaker	**3.2%**

7.1 Revision Analysis

First we explore a preliminary analysis of revision behavior. The average subject made 12 revisions per answer. The average revision was 3 characters long, in that 3 characters would be deleted and then replaced.

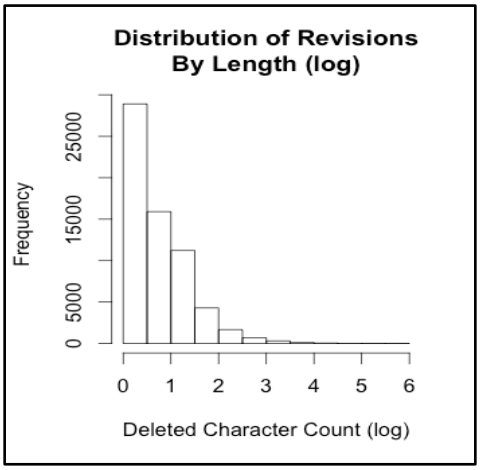


Figure 9. Distribution of revisions

To better understand what happens when typists make revisions, we identified an incomplete taxonomy for 6 types of revisions.

1. Transposition
    a. Switch two letters
    b. INQIURED → INQUIRED
    c. Made up 12% of revisions
2. Fat Finger
    a. Finger accidentally hits neighboring key
    b. AGTER → AFTER
    c. Made up 10% of revisions
3. Doubling Error
    a. "Doubling schema," which tells our finger to double a certain letter, gets attached to the wrong letter
    b. SMAALER → SMALLER
    c. Made up 1% of revisions
4. Early Finger
    a. A finger is activated a few keystrokes too early
    b. POERSON → PERSON
    c. Made up 12% of revisions
5. Word Change
    a. Different Lexical Choice
    b. GRAPH → CELLS
    c. Made up 10% of revisions
6. Unchanged
    a. Revision is the same as deleted text
    b. Made up 3% of revisions
7. Uncategorized
    a. Motivation for revision was unclear from context
    b. Made up 53% of revisions

Splitting up revisions by typing fluency and language nativeness produced a handful of interesting results. Non-fluent typists, referred to as "visual typists" because they rely on looking at their fingers when they type, produce 8% more fat finger errors. However, they produce 6% fewer early finger errors, perhaps because the slower typing rate prevents signals from overlapping from keystroke to keystroke. Non-native English speakers produce 7% fewer "unchanged" errors, where they replace a text with the same text. Perhaps a more limited vocabulary prevents indecision in lexical choice.

Table 18. Impact of revision types

Revision Type	% Native English	% Visual Typist
Transposition	80% (+1%)	25% (-4%)
Unchanged	72% (-7%)	32% (+3%)
Fat Finger	76% (-3%)	37% (+8%)
Word Change	79% (0%)	27% (-2%)
Doubling Error	75% (-4%)	31% (+2%)

Early Finger	80% (+1%)	23% (-6%)

Also Incorporating revisions into authentication experiments marginally improved results. By eliminating digraphs that were ultimately deleted (reparanda) we improved digraph-based authentication EER from 0.080 to 0.072. These results seem to suggest that keystrokes produced as reperanda are ultimately less reliable. In other words, a subject is less "like themselves" and not following their own normative behavior during these reparanda bursts. By identifying these bursts we can eliminate noisy, outlier keystrokes.

7.2 Feature Pruning

To improve authentication results, 4 pruning methodologies were implemented, 1) Minimum-maximum - If one subject has an observation count above the pruning threshold, then that feature is included for all subjects, 2) Maximum-minimum - If all subjects have observation counts above the pruning threshold, then the feature is included, 3) Top count - Observation counts are tallied across all subjects. The features with cumulative counts above a threshold are included, 4) Z-score Pruning - The mean of the absolute values of the z-scores of observations for a feature are averaged across all subjects. If the mean z-score is less than the pruning threshold, the feature is included

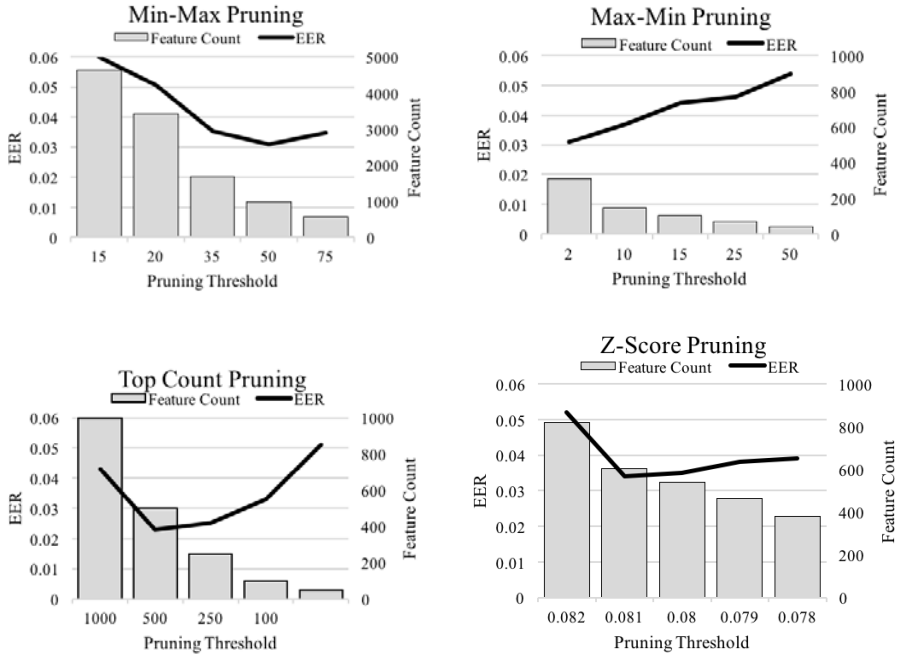


Figure 10. Impact of different pruning schemes

From this analysis we are able to draw a number of conclusions. Maximum-minimum and minimum-maximum pruning are not flexible enough. Minimum-maximum pruning allows in too many features, while minimum-maximum pruning eliminates too many features. A second attempt was made to prune features depending on where they occurred relative to word boundaries. Previous studies have hypothesized that intra-word typing rate, i.e. keystrokes produced within word boundaries, provide a more reliable metric for typing proficiency. In the chart below, we looked at the pauses before each keystroke in every 5-character word. As can be seen, intervals are relatively constant within a word, but lengthened at the word edges.

We hypothesized that this could extend to authentication and provide a more reliable signal for authentication. These findings were partially supported by the fact that variance of timing within a word is significantly less than variance surrounding spaces and punctuation

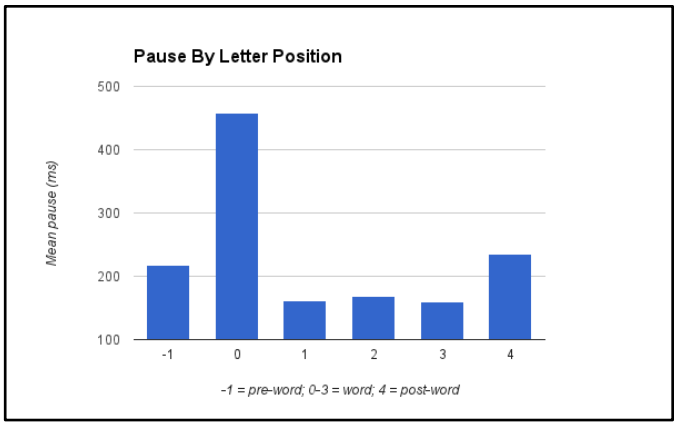
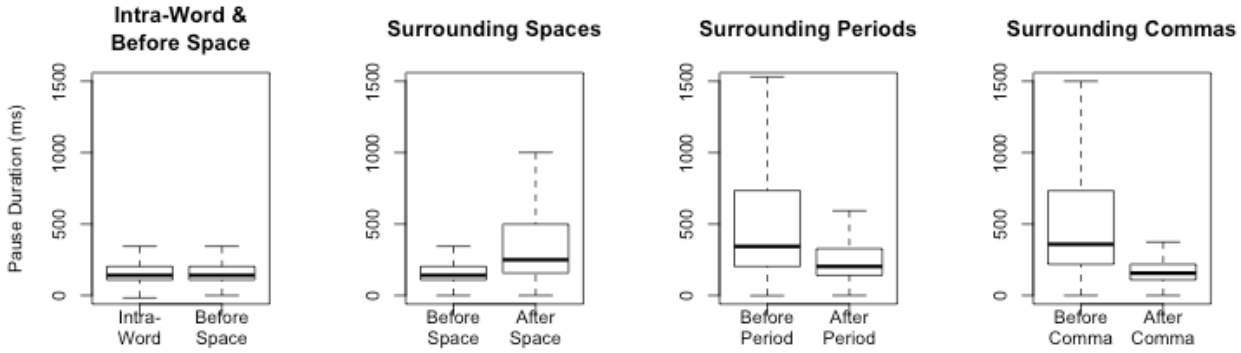


Figure 11. Pause by letter position

However, removing keystrokes that were produced in high variance environments such as following spaces or preceding punctuation did not improve experimental results. It is possible that variance is a necessary and important component of a subject's typing behavior.



.

Figure 12. Duration of pause at different location

7.2.1.1 Analysis of Multiword Expressions

Here we explore the timing of the production of Multi-Word Expressions (MWEs). We measured pauses in between words and found a significant difference between pauses between words that occurred within MWEs and pauses that occurred outside of MWEs

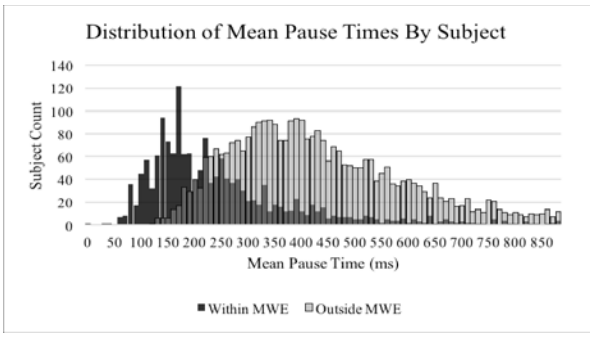


Figure 13. Distribution of mean pause

The mean pause between words within a MWE was not only shorter but was more concentrated around the mean

As an added variable, we investigated how MWE production is affected by manipulating the cognitive complexity of the overall task. Different prompts were associated with different levels of cognitive demand. For instance a simple question might ask the subject to recall a recent experience while a more demanding task could ask a subject to critically analyze a complex subject.

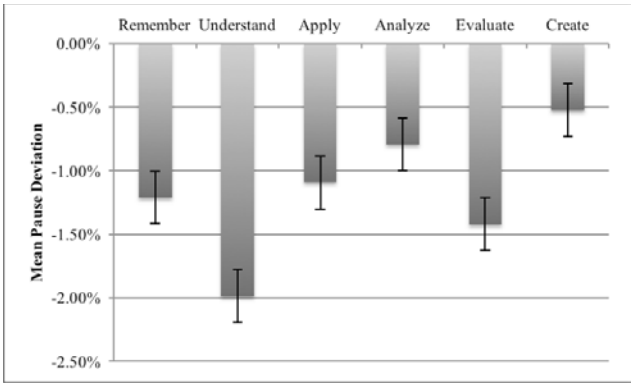


Figure 14. Similarity between a free expression and an MWE

The chart above maps the similarity between a free expression and an MWE. As tasks change, MWE production becomes significantly more or less similar to free expression production. We hypothesize that this change is due to an interaction between lexical retrieval and the cognitive module that controls overall executive function.

Overall, our experiments demonstrated the relevance of linguistic context in classifying keystroke patterns, whether for user authentication or identifying the characteristics of a population. By situating an n-graph within the context of the linguistic structure it was produced within, the set of observations that make up each feature will be more meaningful.

8 IMPACT OF KEYSTROKE FEATURE TRANSFORMATIONS

8.1 Motivation

The performance of a biometric user authentication system considerably relies on the types of features used. One aspect of a biometric feature that is central to its behavior and classification performance is its underlying statistical distribution. From a theoretical perspective, measures such as the feature entropy and the Bayes classification error — widely used by feature quality evaluation schemes (e.g., see [31] [32]) — directly or indirectly depend on the underlying distributions of the features. We studied the question of whether simple feature transformations that manipulate the underlying statistical distributions of keystroke features could have a significant impact on classifier error rates.

8.2 Assumptions

We assume that the independent keystroke features: key hold time of a typed letter x, KHTx, and the key interval time between two consecutively typed letters x and y, KITxy , follow normal distribution. Our assumption of normality is motivated by three factors: (1) the normal distribution, for a given mean and standard deviation, has the maximum entropy among all continuous distributions [37]. Therefore, the assumption of normality leaves us with the largest possible uncertainty (entropy) over the independent features. In other words, the normality assumption ensures that the assumed probability distribution is consistent with the known constraints such as the mean and variance of the independent features, but (maximally) avoids all unknown biases over the features; (2) by the Central Limit Theorem of statistical theory, given a large number of samples, most distributions tend to follow normal distribution [34]; and (3) previous studies [35, 36, 37] on user authentication using keystroke dynamics show that authentication systems performed well when the key hold and key interval times were assumed to follow normal distribution.

Based on the normality assumption on the independent features, we give the distributions of the derived features, the proofs of which can be found in [38]. Since KHT and KIT are independent of each other, they could be visualized as two vectors which are orthogonal to each other. Leveraging the analogy to a right angled triangle, we use the term *hypotenuse* to refer to features such as (3) and (4) (see below) in our discussions. Following the same analogy, we also refer to features taking a ratio of two sides as angle features.

8.3 Derived Features

The following are the features that we derived from the atomic features.

1. Key Press Latency (KPL): between two consecutively typed letters x and y, denoted as KPLxy, is the sum of the key hold time of the first letter x and the key interval time of the letters x and y.

2. Key Release Latency (KRL): between two consecutively typed letters x and y, denoted as KRLxy, is the sum of the key interval time of the letters x and y and the key hold time of the second letter y.

3. Sxy: The hypotenuse feature Sxy between any two consecutively typed letters x and y is given by Sxy=$\sqrt{\left(\mathrm{KPL_{xy}}\right)^2 + (\mathrm{KHT_y})^2}$, where KPLxy is the key press latency between the letters x and y and KHTy is the key hold time of the letter y.

4. Rxy: The hypotenuse feature Rxy between any two consecutively typed letters x and y is given by Rxy=$\sqrt{(\mathrm{KPL_{xy}})^2 + (\mathrm{KHT_x} - \mathrm{KHT_y})^2}$, where KPLxy is the key press latency between letters x and y, KHTx is the key hold time of x, and KHTy is the key hold time of y.

5. Axy: The angle feature Axy between any two consecutively typed letters x and y is given by Axy = $\tan^{-1}\left(\frac{KHT_y}{KPL_{xy}}\right)$, where KHTy is the key hold time of the letter y and KPLxy is the key press latency between the letters x and y.

6. Txy: The angle feature Txy between any two consecutively typed letters x and y is given by Txy =$\tan^{-1}\left(\frac{(KHT_y - KHT_x)}{KPL_{xy}}\right)$, where KHTy is the key hold time of the letter y, KHTx is the key hold time of the letter x, KPLxy is the key press latency between the letters x and y.

7. Nxy: The ratio feature Nxy between any two consecutively typed letters x and y is given by Nxy = $\left(\frac{KHT_y}{KPL_{xy}}\right)$, where KHTy is the key hold time of the letter y and KPLxy is the key press latency between the letters x and y.

8. Vxy: The ratio feature Vxy between any two consecutively typed features x and y is given by Vxy = $\left(\frac{KHT_y - KHT_x}{KPL_{xy}}\right)$, where KHTx is the key hold time of the letter x, KHTy is the key hold time of the letter y, and KPLxy is the key press latency between the letters x and y.

8.4 Distribution of Derived Features

The key press latency between two consecutively typed letters x and y, KPLxy = KHTx + KITxy follows a *normal distribution* under the assumption that KHTx and KITxy are independent normal random variables. Under the same assumption, the key release latency between two consecutively typed letters x and y, KRLxy = KHTy + KITxy follows normal distribution. The hypotenuse feature Sxy between two consecutively typed letters x and y follows a *Rayleigh distribution* with parameter $\sigma = 1$ when KHTy and KPLxy are standardized to zero mean and unit variance. Similarly, the hypotenuse feature rxy between two consecutively typed letters x and y follows a Rayleigh distribution with $\sigma = 1$ when (KHTx − KHTy) and KPLxy are standardized to zero mean and unit variance.

The angle feature Axy between two consecutively typed letters x and y follows a uniform distribution in the interval (0, 2π) when KHTy and KPLxy are standardized to zero mean and unit variance. Similarly, the angle feature Txy between two consecutively typed letters x and y follows a *uniform distribution* in (0, 2π) when (KHTx − KHTy) and KPLxy are standardized to zero mean and unit variance. The ratio feature Nxy between two consecutively typed letters x and y follows a standard *Cauchy distribution* when KHTy and KPLxy are standardized to zero mean and unit variance. Similarly, the ratio feature Vxy between two consecutively typed letters x and y follows a standard Cauchy distribution when (KHTx − KHTy) and KPLxy are standardized to zero mean and unit variance.

Table 19. Keystroke Entropy of Independent and Derived Features

Feature	Distribution	Parameters of pdf	A measure of differential entropy in nats
KHT	Standard Normal	$\mu = 0, \sigma = 1$	$0.5 \ln(2\pi e) = 1.42$
KIT, KPL, KRL	Standard Normal	$\mu = 0, \sigma = 1$	$0.5 \ln(2\pi e) = 1.42$
S, R	Rayleigh	$\sigma = 1$	$1 + \ln(1/\sqrt{2}) + 0.2886 = 0.94$
A, T	Uniform	$a = 0, b = 2\pi$	$\ln(2\pi) = 1.84$
N, V	Standard Cauchy	$\lambda = 1$	$\ln(4\pi) = 2.53$

We define the keystroke feature entropy measure as:

$$\sum_{j=1}^{n}\left(-\int_{x_j} p(x_j|u_i)\, log(x_j|u_i)\right)$$

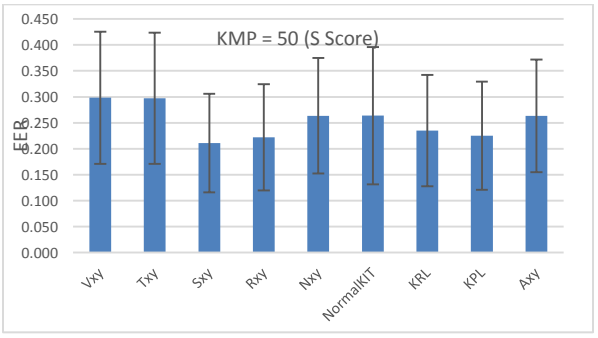
where X = x = (x1, x2, · · · , xn) is a n dimensional continuous feature vector, and U = (u1, u2, · · · , um) represents m different users. Assume that the probability of a feature x_j given a user u_i, $p(x_j|u_i)$, originates from some parametric probability distribution. Let the density of the parametric distribution be f_Y (y), where Y is the support set of the random variable y. Since p(xj|ui) originates from f_Y (y), then $x_j \in Y$. Then, the term $\int_{x_j} p(x_j|u_i)\, log(x_j|u_i)$ in the keystroke feature entropy measure is the same as differential entropy function of f_Y (y), given by $h(y) = -\int_Y f(y) \log(f(y))\, dy$. Therefore, keystroke feature entropy measure of the features can be theoretically calculated using the differential entropy functions of the features.

Table 19 gives the keystroke feature entropy values of the independent and derived keystroke features. Without loss of generality, the independent features KHT and KIT that follow the normal distribution are standardized. Similarly, the derived features KP L and KRL that follow the normal distribution are standardized. Because KHT and KPL are standardized, the hypotenuse features S and R follow Rayleigh distribution with σ = 1, the angle features A and T follow uniform distribution in the interval (0, 2π), and the ratio features N and V follow standard Cauchy distribution. The column 'Differential Entropy Function' gives the differential entropy functions of the keystroke features. The general forms of the differential functions are given in

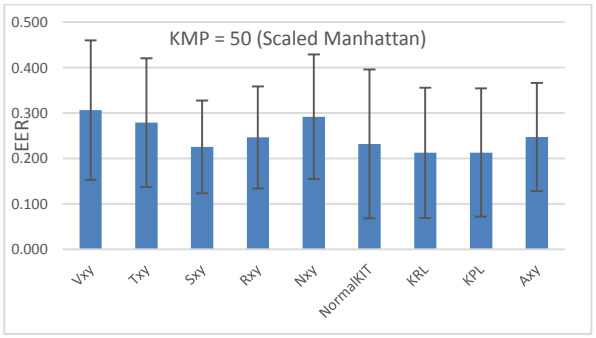
[27]. Table 19 shows that the derived features S and R have the minimum keystroke feature entropy values, indicating that the bound on Bayes error is minimum for S and R among all the features.

8.5 Results

We carried out an empirical investigation to see how the observed theoretical behavior might apply to a real authentication system. We ran experiments with the various independent and derived features to study the error rates seen with different verification algorithms. Figure 15 summarizes these results. For all matching pairs (50 up to 350) studied, the features S and R appear to at least perform as well as the widely used KIT feature for both the Scaled Manhattan and Z Score verifiers (i.e., in terms of EER values). In the majority of cases these features actually perform better than the KIT. These results to a good extent agree with the theoretical results and suggest that the derived features might have promise.

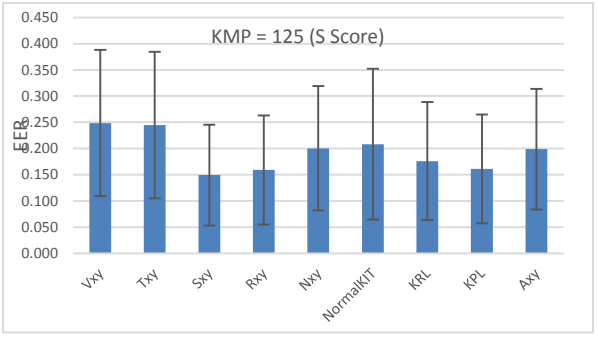


(a) Score calculated using Z Score verifier

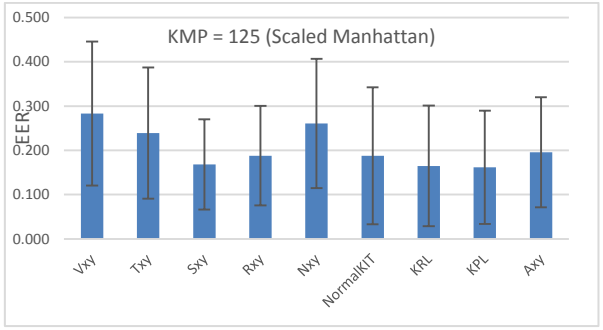


(b) Score calculated using Scaled Manhattan Distance verifier

Key Matching Pairs value of 50

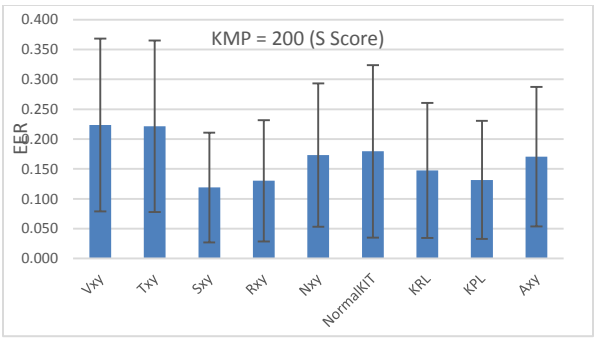


(c) Score calculated using Z Score verifier

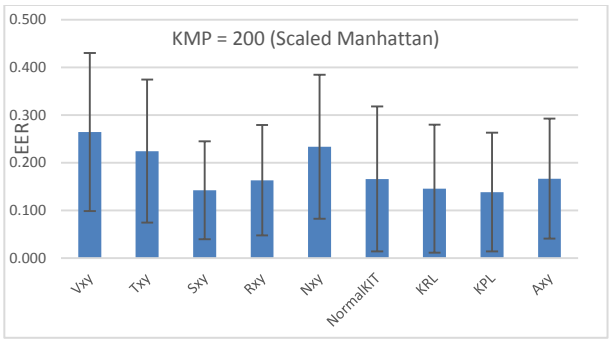


(d) Score calculated using Scaled Manhattan Distance verifier

Key Matching Pairs value of 125

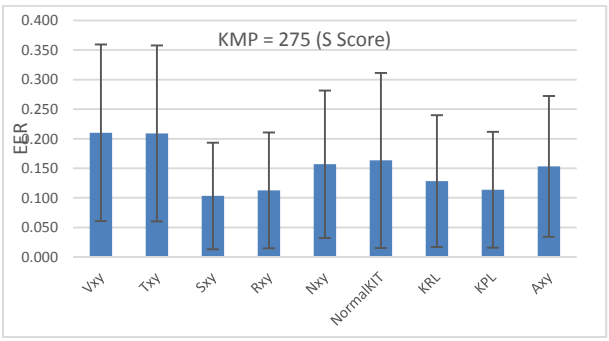


(e) Score calculated using Z Score verifier

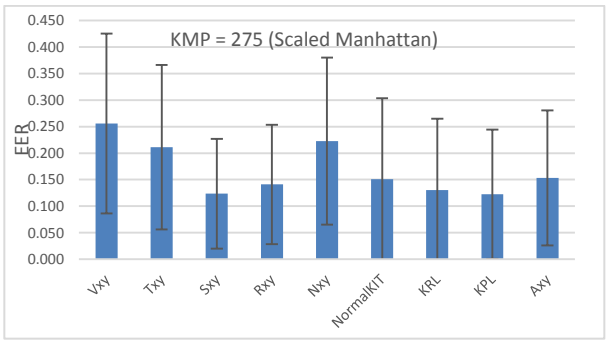


(f) Score calculated using Scaled Manhattan Distance verifier

Key Matching Pairs value of 200

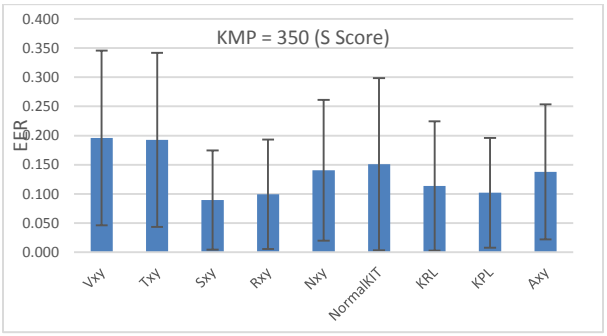


(g) Score calculated using Z Score verifier

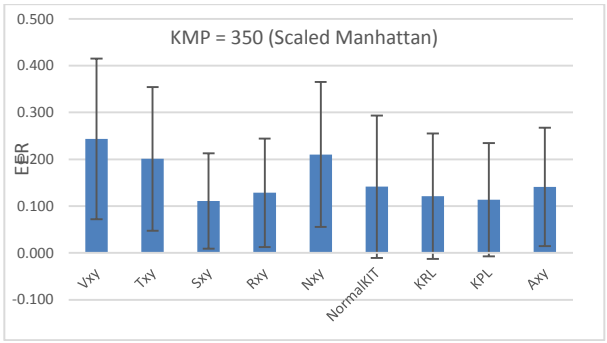


(h) Score calculated using Scaled Manhattan Distance verifier

Key Matching Pairs value of 275



(i) Score calculated using Z Score verifier



(j) Score calculated using Scaled Manhattan Distance verifier

Key Matching Pairs value of 350

Figure 15: EER for different methods and different key matching pairs values

9 POPULATION ATTACK

We conducted population level attacks and investigating the impact of statistical blocking parameters. We define the concept of the population attack as taking a large user population and generating test vectors from users who have not been trained in order to attack users who have been trained. In our experiment, 352 users were used as the attacker set and 486 users have been attacked. The EERs are listed in the Tables below. Features studied include unigraph keyhold feature (KH), the digraph feature (DN), and the pausality feature (PP), and the authentication window ranged from 30 seconds to 150 seconds at 30 second intervals (shown on the top row).

Table 20. Population level attack on users with age <=20 years

<=20 years	30s	60s	90s	120s	150s
KH	0.193	0.143	0.135	0.085	0.022
DN	0.124	0.093	0.072	0.030	0.059
PP	0.330	0.193	0.124	0.080	0.041

Table 21. Population level attack on users with age >=25 years

>=25 years	30s	60s	90s	120s	150s
KH	0.151	0.111	0.127	0.056	0.016
DN	0.159	0.127	0.103	0.056	0.056
PP	0.310	0.167	0.135	0.087	0.032

Tables 22 and 23 below show the population level attack results with the impact of the statistical blocking parameter of average hours spent typing.

Table 22. Population level attack results on users with average hours spent typing/day <=3

. <=3 hours	30s	60s	90s	120s	150s
KH	0.175	0.138	0.149	0.067	0.034
DN	0.131	0.108	0.075	0.041	0.052
PP	0.343	0.205	0.119	0.063	0.034

Table 23. Population level attack on users with average hours spent typing / day >= 6

>= 6 hours	30s	60s	90s	120s	150s
KH	0.185	0.116	0.111	0.083	0.019
DN	0.111	0.083	0.060	0.037	0.042
PP	0.301	0.176	0.125	0.088	0.019

Next, Tables 24 and 25 below show the population level attack results with the impact of the statistical blocking parameter of consciousness. Users can be classified as either a conscious typist, or a non-conscious typist.

Table 24. Population level attack on users who claim themselves as conscious typists

Consciousness	30s	60s	90s	120s	150s
KH	0.169	0.125	0.128	0.078	0.024
DN	0.145	0.108	0.081	0.047	0.057
PP	0.287	0.199	0.128	0.078	0.034

Table 25. Population level attack on users who claim themselves as non-conscious typists

Non-Conscious	30s	60s	90s	120s	150s
KH	0.186	0.135	0.132	0.076	0.014
DN	0.120	0.098	0.067	0.025	0.048
PP	0.321	0.171	0.127	0.082	0.033

Next, tables 26 and 27 below, show the population level attack results with the impact of the statistical blocking parameter of first language. Users have specified if their first language is either English or non-English.

Table 26. Population level attack on users who specified their first language as English.

English	30s	60s	90s	120s	150s
KH	0.190	0.132	0.132	0.074	0.021
DN	0.122	0.093	0.062	0.034	0.058
PP	0.308	0.180	0.118	0.077	0.031

Table 27. Population level attack on users who specified their first language as Non English

Non-English	30s	60s	90s	120s	150s
KH	0.156	0.125	0.125	0.063	0.013
DN	0.156	0.125	0.100	0.050	0.044
PP	0.331	0.213	0.156	0.094	0.038

Finally, Tables 28 and 29 show the population level attack results with the impact of the statistical blocking parameter of prior training. To review, prior training is defined as a user who has taken a formal typing class prior to taking the survey for our research project.

Table 28. Population level attack on users who specified that they have had prior training.

Yes	30s	60s	90s	120s	150s
KH	0.207	0.142	0.140	0.071	0.011
DN	0.118	0.096	0.060	0.018	0.044
PP	0.318	0.182	0.127	0.082	0.047

Table 29. Population level attack on users who specified that they have not have prior training

No	30s	60s	90s	120s	150s
KH	0.156	0.118	0.107	0.072	0.022
DN	0.140	0.110	0.079	0.050	0.055
PP	0.294	0.186	0.127	0.077	0.029

10 IMPACT OF CLASSIFICATION THRESHOLDS

The classification threshold is one of the key parameters of an authentication threshold that need to be very finely tuned for good performance. There exist very many approaches to setting the classification threshold of a biometric authentication system, however, the vast majority of these methods have never been evaluated for keystroke authentication (most studies report results based on the EER threshold, which despite being sufficient for performance evaluation purposes in a research setting is typically not used in real systems where more sophisticated trade-offs between the FAR and FRR have to be made). To guide design decisions for the prototype designed by the LaTech team, we studied several threshold setting methods to not only determine which ones perform best, but also get insights into why some of them do (or don't) perform well from a keystroke authentication standpoint.

10.1 Comparing performance of threshold-setting methods

We implemented eight methods for threshold-setting that are well established in the literature and evaluated them on our data. These methods are: (1) the equal error rate method, (2) Gauss method, (3) 3σ method, (4) Furui method, (5) CAVE-1, (6) CAVE-2, (7) CAVE-3 and (8) Ke Chen's method. Implementation details of these methods can be found in [21] and the papers citeds therein, however, we briefly discuss the methods here for completeness. We use $\mu \; and \; \sigma$ to respectively represent the mean and standard deviation of the genuine scores and $\bar{\mu} \; and \; \bar{\sigma}$ to respectively represent the mean and standard deviation of the impostor scores. T_s represents the classification threshold.

Equal error threshold (e.e.t):

Equal Error Rate (EER) is approximated through use of the corresponding Half Total Error Rate (HTER) on condition that the absolute value of the difference between False Acceptance Rate

and False Rejection Rate is minimal. Equal error threshold is the value of the threshold where the EER occurs.

Gauss method:

The method assumes that the distributions of genuine scores and imposter scores are Gaussian. The decision threshold can be achieved as shown below, where $G(x|\mu,\sigma)\ and\ G(x|\bar{\mu},\bar{\sigma})$ are the genuine and impostor score distributions with the parameters μ,σ and $\bar{\mu},\bar{\sigma}$ respectively.

$$T_s = arg_x(\ G(x|\mu,\sigma) = G(x|\bar{\mu},\bar{\sigma})\ )$$

3σ method:

This method also makes the Gaussian assumption and is based on the following property: 99.7% of all the samples drawn from Gauss distribution should be located within only the interval $[\tilde{\mu} - 3\tilde{\sigma}, \tilde{\mu} + 3\tilde{\sigma}]$. The threshold is computed based on the following expression.

$$T_s = \begin{cases} \mu - 3\sigma & if\ \mu - 3\sigma > \bar{\mu}\ + 3\bar{\sigma} \\ (\mu\bar{\sigma} + \bar{\mu}\sigma)/(\sigma + \bar{\sigma}) & otherwise \end{cases}$$

Furui method:

This method assumes that genuine scores are unreliable (due to the fact that genuine scores tend to be very few, in fact much fewer than the impostor scores). The method sets the threshold using only the imposter scores. The threshold is decided by the following equation:

$$T_s = \alpha(\bar{\mu} + \bar{\sigma}) + \beta$$

The parameters α and β are estimated as follows: For each value of α and β, the threshold is estimated by the above equation and then used to achieve an HTER. The optimal parameters are obtained when minimal HTER is obtained.

CAVE-1:

A similar method with Furui method except that it uses a linear combination of the estimates of the means of the genuine and imposter scores. γ can be computed by parameter estimation using an approach similar to the one used with the Furui method.

$$T_s = \gamma\mu + (1 - \gamma)\bar{\mu}$$

CAVE-2:

Based on the Bayesian threshold, a user-independent correction, δ, is introduced to adjust the estimate of the mean on genuine scores only.

$$T_s = arg_x\left(\frac{G(x|\hat{\mu},\hat{\sigma})}{G(x|\bar{\mu},\bar{\sigma})} = T_B\right) \quad \text{where } \hat{\mu} = \mu - \delta\ and\ \hat{\sigma} = \sigma\ ,$$

$$T_B \; is \; Bayesian \; Threshold$$

CAVE-3:

Because the user-independent threshold setting method demands a good amount of data, a user-dependent adjustment is made to improve CAVE-1. Both the user-independent threshold TSI and parameter η are optimized on a registration population.

$$T_s = T_{SI} + \eta(\bar{\mu} - \mu)$$

Ke Chen's method:

The threshold is computed using the equation below, where a and b are two user-independent parameters and optimized on a registration population.

$$T_s = b(\bar{\mu} + a\bar{\sigma}) + (1 - b)\mu$$

This method has a pruning step which is implemented as follows:

1. For a set of scores, find $\mu \; and \; \sigma$.
2. Define $d_i = |x_i - \mu|$, find maximum d_i
3. Eliminate that element
4. Re-estimate μ
5. Repeat 2-4 until all x_i satisfies $d_i < K\sigma$ (K simply use 2, how to find a good enough K is still an open problem)
6. Then we got a data set with all normal data.

For performance evaluation we used a dataset of 486 users. We used the most frequent 100 digraphs and used only the KIT feature. KITs greater than 1000 ms were filtered out as outliers. We used two verifiers which are very widely used in keystroke authentication, namely, the Scaled Manhattan verifier and the Z-score verifier.

Figure 16 to Figure 21 show the performance of the various methods. One section of data was used for training, while the other was used for testing. These sections of data were collected on different days, so the results reported here capture some variability of typing behavior between days. We used different window sizes for testing, namely 50, 100 and 150 pairs. As an example, a window size of 50 pairs means that a user's entire testing dataset is split into segments of 50 character-pairs, with each segment being used to compute a match score against the user's template. Impostor samples were chosen as follows: Out of 485 possible impostors, an impostor was chosen at random, and 50 pairs randomly chosen from that impostor. This process was repeated 50 times to generate 50 impostor scores per user.

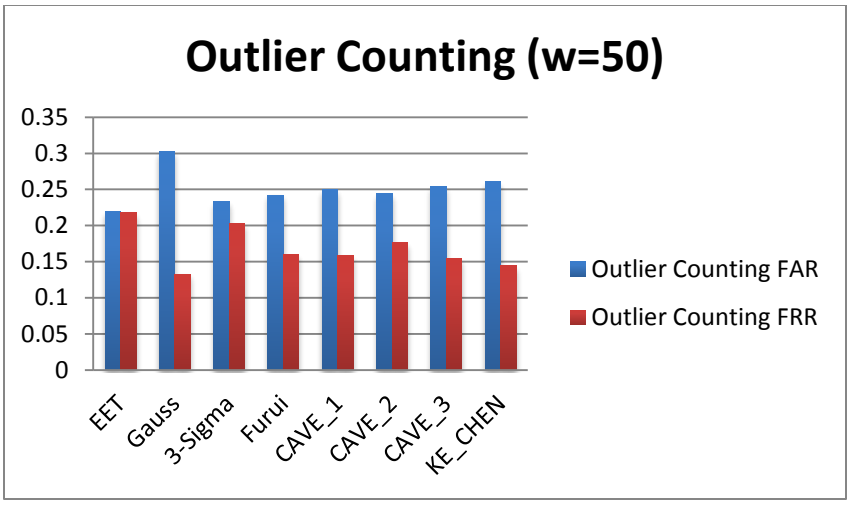


Figure 16: Performance of Z-score Classifier for a window size of 50 pairs.

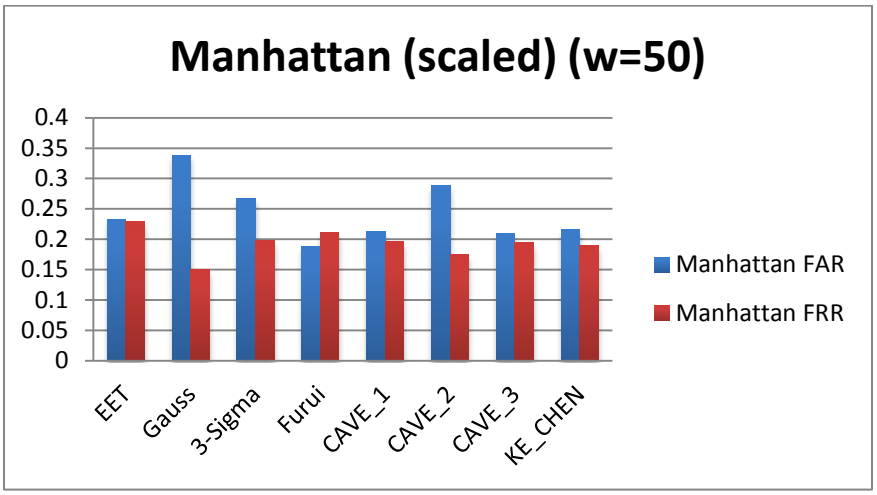


Figure 17: Performance of Scaled Manhattan Classifier for a window size of 50 pairs.

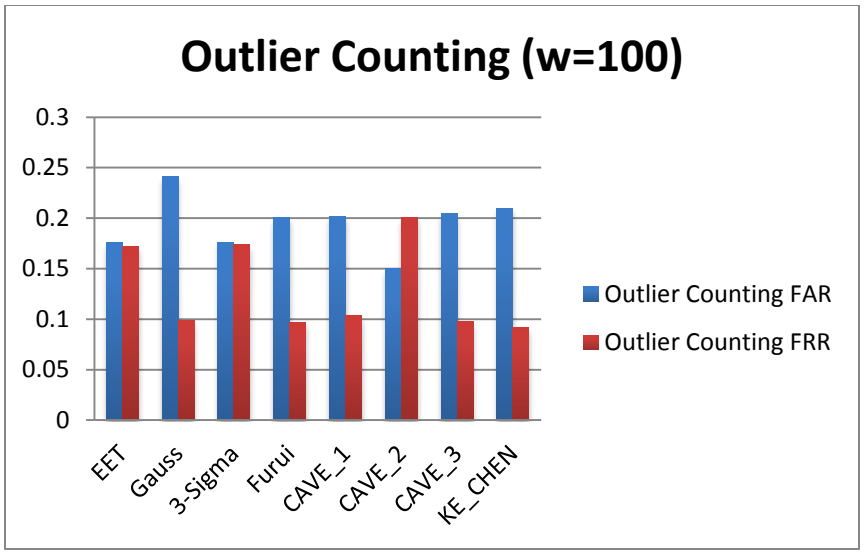


Figure 18: Performance of Z-score Classifier for a window size of 100 pairs.

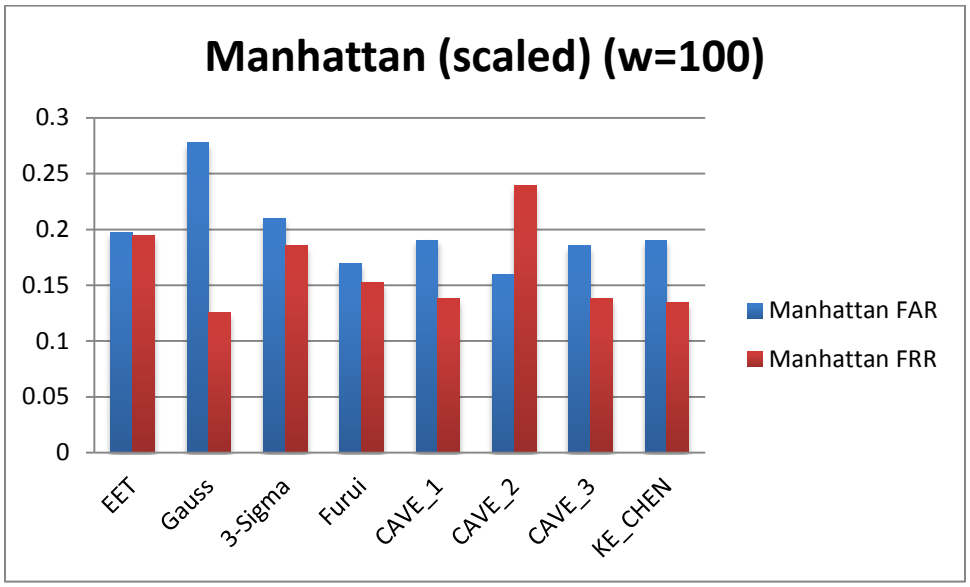


Figure 19: Performance of Scaled Manhattan Classifier for a window size of 100 pairs.

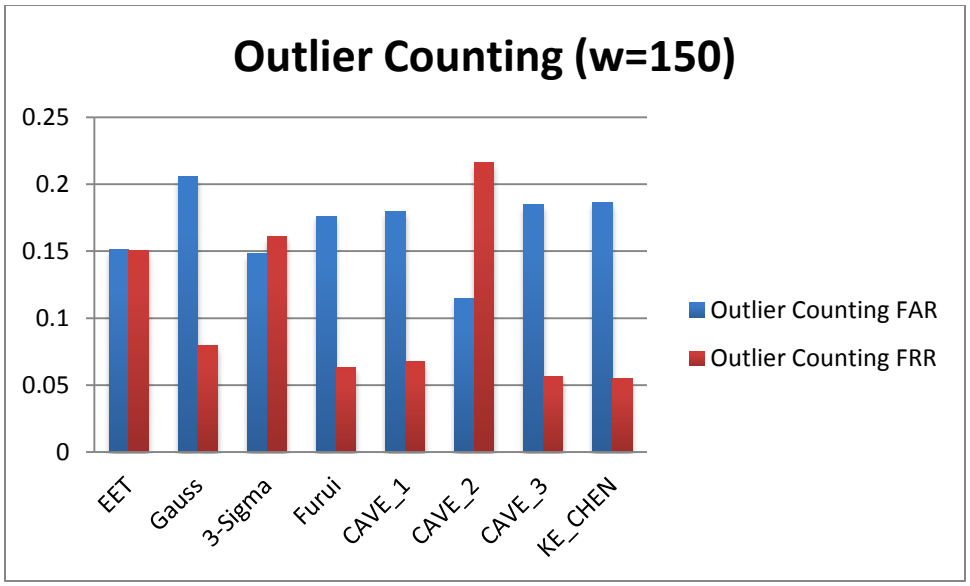


Figure 20: Performance of Z-score Classifier for a window size of 150 pairs.

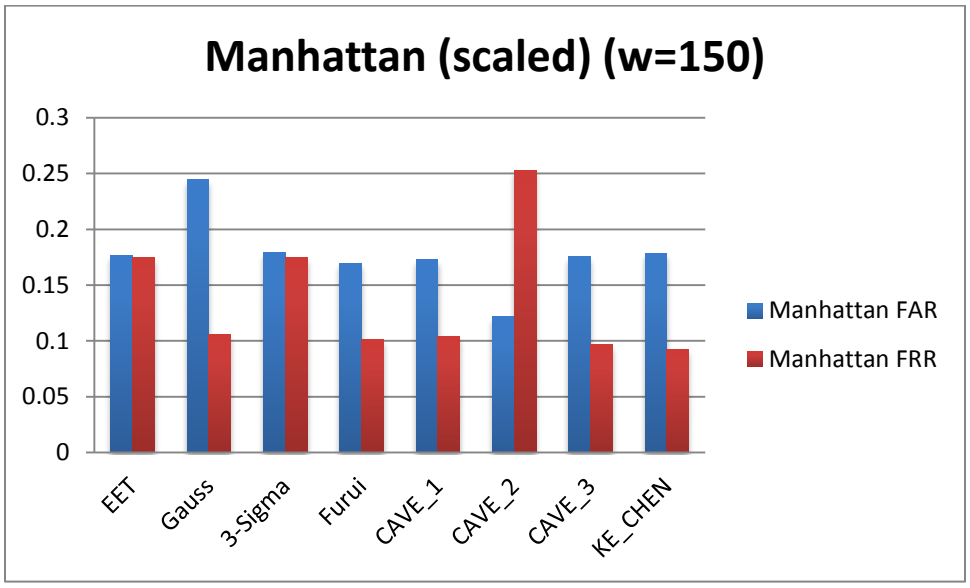


Figure 21: Performance of Scaled Manhattan Classifier for a window size of 150 pairs.

Observations about the methods:

(1) Considering the mean of the FAR and FRR for each method, the two methods which make the Gaussian assumption for the most part perform worse than those which don't (i.e., the Furui, CAVE_1, CAVE_3 and Ke-chen methods). This trend is especially apparent for the (largest) window size, w=150.

(2) Irrespective of threshold setting method, the window size used seems to have a significant effect on the error rates. While the average error rate is around 0.22 for w=50, it comes down to around 0.17 for w=150. A similar trend is seen for all the other

methods. This observation suggests that the window size might be a more critical parameter than the threshold setting approach.

10.2 Root-cause Analysis

We carried out a root-cause analysis to try to understand why the different methods performed the way they did. In general, if one can point a finger to a reason why a certain method performs well (or poorly), it could be possible to apply it selectively under conditions in which it might work well. On the other hand if a method always performs poorly (irrespective of parameter settings or specific statistical properties of data), then such a method could be classified as being inappropriate for our application.

10.2.1 The Biometric "Animals"

The first investigation we carried out was whether the performance of a given threshold-setting approach depended on the type of user under consideration (i.e., a user whose features were consistent across authentication attempts Vs a user who depicted significant intra-user variations in the features). For this analysis, we leveraged the biometric menagerie [28,30], a user categorization method that assigns users labels of different animals depending on their classification performance. We focus on the classical form of the menagerie which uses three types of animals, namely, goats, lamb and sheep. The three user categories are described as below in the seminal Doddington Zoo" paper [24]:

- Goats are users who are intrinsically difficult to recognize and they tend to adversely degrade the performance by increasing FRR.
- Lambs are users whose biometric feature set overlaps significantly with other users in the database thereby contributing to a high FAR.
- Sheep are well-behaved users exhibiting low FRRs and whose feature sets are well separated from other users in the database.

Because the precise score thresholds for making these categorizations vary for different biometric modalities, we use the thresholds in [30] where the biometric menagerie was studied from the perspective of a keystroke authentication system.

Figure 22 and Figure 23 show the performance of the different 'animals' when the Scaled Manhattan and Outlier counting verifiers were respectively used for classification. Results are based on the full set of 486 users. Performance is reported in terms of the HTER (Y-axis), which is the mean of the FAR and FRR for a given threshold-setting approach. The HTER is computed for each of the three animal categories, and also computed for the full population (See line labeled TOTAL) for comparison. Observe that irrespective of the kind of animal, the trend in HTERs for the most part remains the same. A similar trend is seen when verifier fusion is used (Figure 24). In all cases there is some kind of peak around either of the first three methods or CAVE-2, and the lowest points being around the Furui, KE-CHEN and CAVE-3 methods. The same trend is seen when the Scaled Manhattan verifier is fused with the Outlier counting verifier. The trend suggests that a user-specific approach to selecting the threshold-setting algorithm might not guarantee any performance benefits.

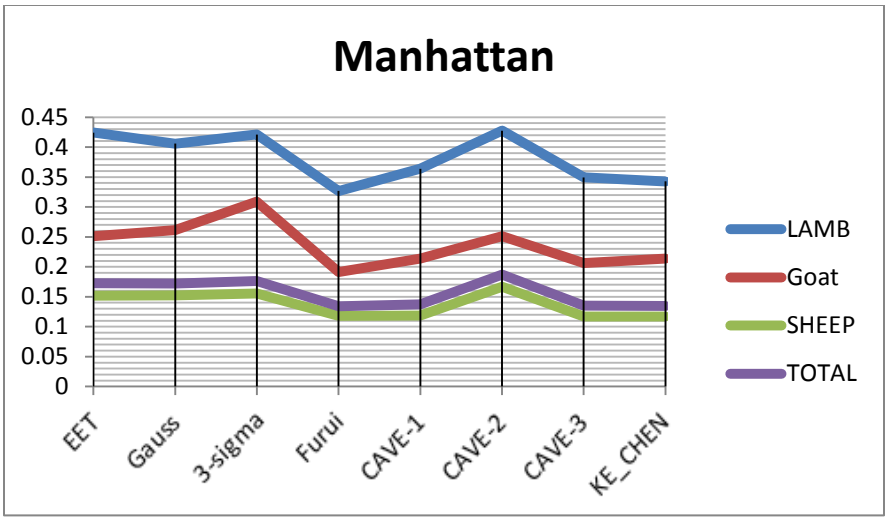


Figure 22: Comparing threshold-setting strategies across the different "animals" when the Scaled Manhattan verifier was used for classification

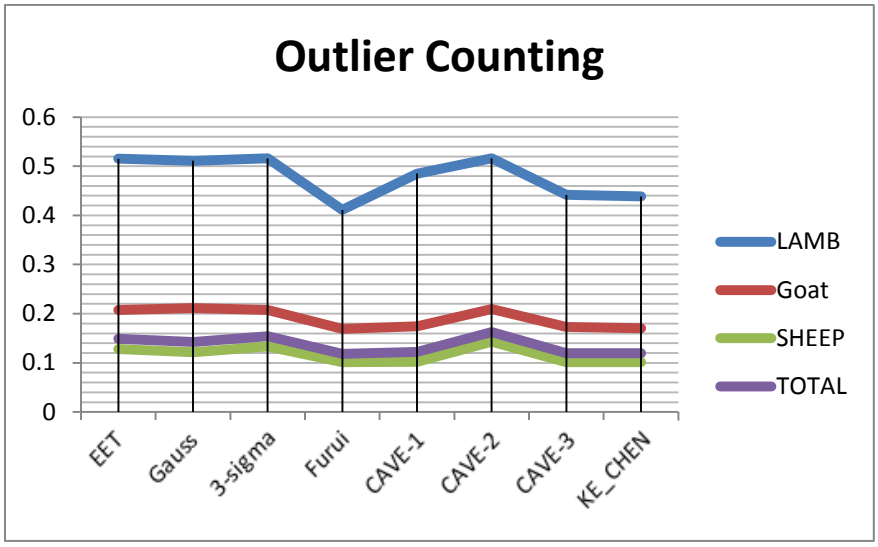


Figure 23: Comparing threshold-setting strategies across the different "animals" when the Outlier Counting verifier was used for classification

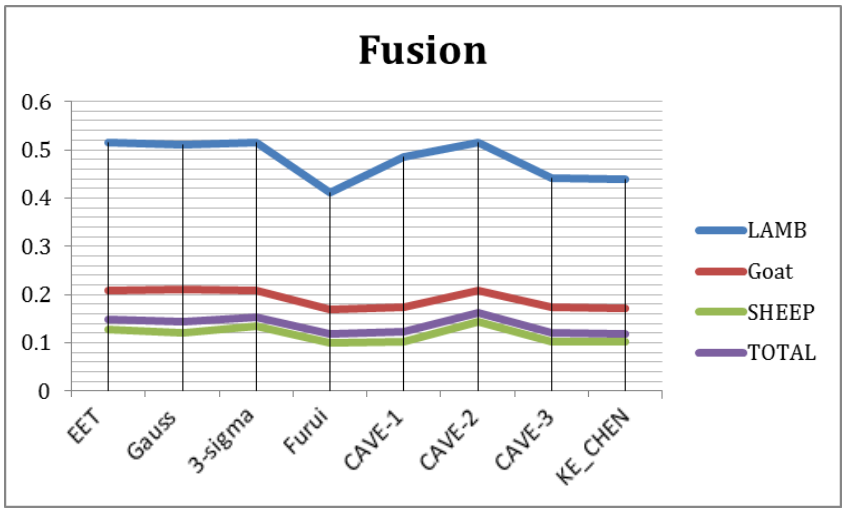


Figure 24: Comparing threshold-setting strategies across the different "animals" when the Outlier Counting verifier was fused with the Scaled Manhattan verifier

10.2.1.1 Random Sampling

To further study how different threshold-setting approaches interact with the different “animals”, we applied a basic form of data resampling (with replacement) in which only subsets of the full population were studied. In particular we randomly selected a set of 81 users 80 times and each time computed the HTER across the different animals and across the full population. The advantage with this approach is that each random set of 80 users simulates a different population. Certain subsets of 80 users may have no overlap at all, while others may have varying extents of overlap. Overall, the mean behavior seen across different sets of 80 users gives a less biased view of the performance trends than a one-shot approach which is based on a single computation carried out on the full dataset.

Figure 25 and Figure 26 summarize the results from this analysis. Observe that the earlier discovered trend for the most part persists. The goats seem to have some variations in absolute values of HTER (not so much the trend), however this difference is likely because we had very few goats from small sub-populations, making their behavior highly variable across sub-sets. Notably, this unclear behavior seen with the goats still does not change the fact that CAVE-2 performs worst, and that EET, Gauss and Sigma are comparable to the CAVE-2 method.

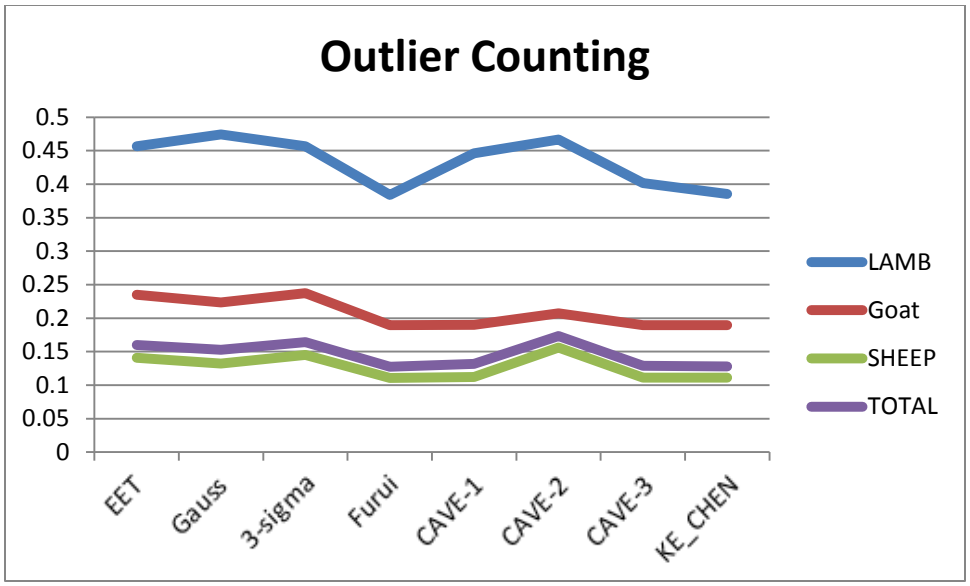


Figure 25: Studying Threshold-setting Methods with the Outlier Counting verifier and User Resampling

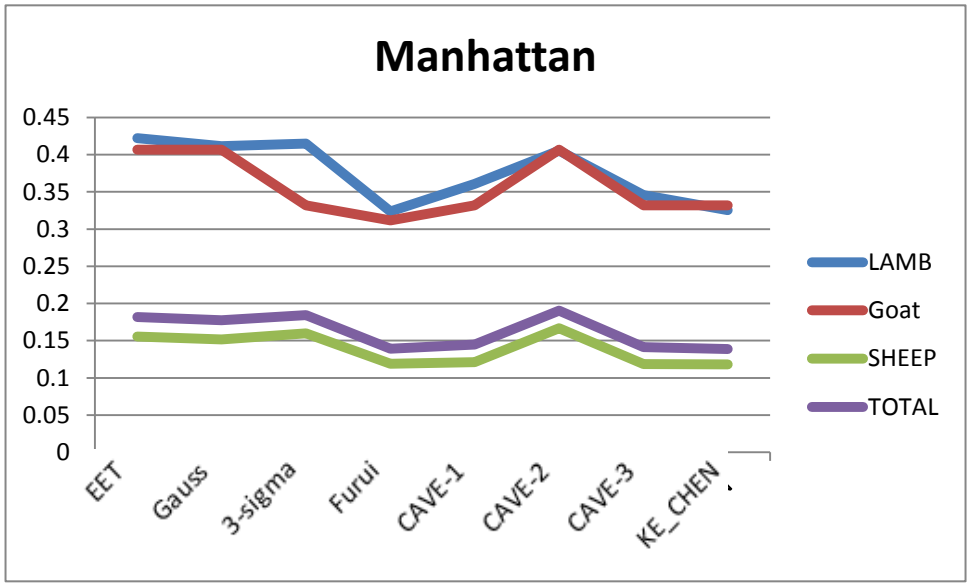


Figure 26: Studying Threshold-setting Methods with the Scaled Manhattan verifier and User Resampling

10.2.2 Statistical Tests of Significance

Here we address the following question: are the differences seen between the methods statistically significant? Since we make no parametric assumptions about the behavior of the HTERs, we use the two-sample Kolmogorov Smirnov test [29] to address this question.

Man	eet	Gauss	Three-sigma	Furui	CAVE-1	CAVE-2	CAVE-3	KE_CHEN
eet		0.018	0.0292	1.64E-21	7.74E-19	4.80E-09	3.75E-20	1.73E-19
Gauss			2.28E-07	3.38E-18	9.49E-16	4.95E-14	2.41E-16	9.49E-16
Three sigma				7.27E-26	6.43E-23	7.16E-06	2.78E-24	1.23E-23
Furui					0.0292	1.26E-30	0.5316	0.666
CAVE-1						2.09E-27	0.3044	0.2201
CAVE-2							3.41E-28	3.41E-28
CAVE-3								0.9968
KE_CHEN								

Figure 27: Results of two-sample KS test on HTERs of various Threshold-setting approaches when the Scaled Manhattan Verifier is used for classification

Out	eet	Gauss	Three-sigma	Furui	CAVE-1	CAVE-2	CAVE-3	KE_CHEN
eet		0.9065	0.1061	2.65E-32	5.41E-29	9.04E-08	1.85E-31	1.85E-31
Gauss			7.08E-02	8.37E-30	2.09E-27	1.31E-08	8.37E-30	1.26E-30
Three sigma				5.02E-34	1.85E-31	7.21E-05	3.70E-33	5.02E-34
Furui					0.0708	1.08E-36	0.9065	0.9968
CAVE-1						8.56E-36	0.3044	0.1061
CAVE-2							8.56E-36	8.56E-36
CAVE-3								0.9729
KE_CHEN								

Figure 28: Results of two-sample KS test on HTERs of various Threshold-setting approaches when the Outlier Counting Verifier is used for classification

Figure 27 and Figure 28 summarize the results (particularly P-values) from these tests. We use alpha=0.05. The pink boxes indicate rejection of the null hypothesis of distribution similarity between a compared pair of verifiers. The most interesting observation here is the consistent trait seen with both verifiers between Furui, CAVE-1 and CAVE-3 and Ke-Chen. Irrespective of verification algorithm used, we are unable to reject the hypothesis that the CAVE-1 method produces an error distribution similar to that of CAVE-3 and Ke-Chen, and that the Furui method produces an error distribution similar to that of CAVE-3 and Ke-Chen.

Having made the comparison between distributions, we proceeded and evaluated whether there were any users whose verification scores (genuine and/or impostor) were Gaussian. This test could help explain the behavior of methods such as 3-sigma and Gauss which inherently rely on the Gaussianity assumption. For this task we used the 1-sample K-S test and at the 5% significance level rejected the hypothesis that scores were Gaussian. This confirmed why the Gaussian-based methods always had higher error rates than the best four methods listed earlier. For our final investigation – the question of whether any method performed better than the others – we applied the Wilcoxon signed rank test and compared each method with the Furui method

(consistently one of the methods with the lowest error rates). These tests could not separate the earlier noted *four* methods, confirming that they are comparable to each other in performance.

11 CONCLUSIONS

We consistently achieve below an HTER of below .02 for single application data collected at Louisiana Tech but experiments with more realistic multi-window, multi-application West Point data show that the authentication may performance significantly drop if application context is not considered.

Another challenge keystroke based authentication encounter is spoof attack. We show that, using population statistics, attacks can be launched with high success rate for single verifier based simple schemes. We overcome this challenge with multi-feature, multi-verifier based composite solution that has a very low imposter pass rate.

Authentication performance somewhat drops with time drift. To address this issue periodical updates of the profile through controlled training may be required.

In the current scheme a classifier makes binary decisions (i.e., genuine or imposter) by comparing the associated scores with its predetermined threshold; it does not consider how close a score to the threshold is. It has been shown that the quality of the classifier decisions can significantly impact the overall decision performance [14].

User Specific Selection of Threshold Schemes and Classifier Weights: In our study a user specific HTER threshold scheme yielded the best HTER in 572 out of 736 cases. However, in 97 cases K-Chen threshold method outperformed the user specific method and in 58 cases the population based HTER method outperformed the user specific method. Additionally, in 9 cases both K-Chen and population based HTER methods outperformed the user-specific method. Given these results, it is possible that selection of threshold computation method on a subject by subject basis can improve overall system accuracy.

In addition, in our current work the classifier decision weights for weighted fusion were set by optimizing the average HTER for all 736 users. Investigating how user-specific weighing of the classifiers impact authentication performance, will be interesting.

12 REFERENCES

[1]F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Transactions on Information Systems Security*, vol. 5, no. 4, pp. 367–397, Nov 2002. [Online]. Available: http://doi.acm.org/10.1145/581271.581272

[2]K. Killourhy and R. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *IEEE/IFIP International Conference on Dependable Systems Networks, 2009. DSN '09.*, June 2009, pp. 125– 134.

[3]T. Shimshon, R. Moskovitch, L. Rokach, and Y. Elovici, "Continuous verification using keystroke dynamics," in *International Conference on Computational Intelligence and Security (CIS), 2010*, Dec 2010, pp. 411–415.

[4]D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information Systems and Security*, vol. 8, no. 3, pp. 312–347, Aug 2005. [Online]. Available: http://doi.acm.org/10.1145/ 1085126.1085129

[5]F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, ser. CCS '97. New York, NY, USA: ACM, 1997, pp. 48–56. [Online]. Available: http://doi.acm.org/10.1145/ 266420.266434

[6]K. Rahman, K. Balagani, and V. Phoha, "Snoop-forge-replay attacks on continuous verification with keystrokes," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 528–541, March 2013.

[7]K. Killourhy and R. Maxion, "Why did my detector do that?!: Predicting keystroke-dynamics error rates," in *Proceedings of the 13th International Conference on Recent Advances in Intrusion Detection*, ser. RAID'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 256–276. [Online]. Available: http://dl.acm.org/citation.cfm?id=1894166.1894184

[8] F. Bergadano, D. Gunetti, and C. Picardi, "Identity verification through dynamic keystroke analysis," Intelligent Data Analysis, vol. 7, no. 5, pp. 469–496, Oct. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1293861.1293866

[9]Y. Sheng, V. Phoha, and S. Rovnyak, "A parallel decision tree-based method for user authentication based on keystroke patterns," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 35, no. 4, pp. 826–833, Aug 2005.

[10] K. Niinuma, U. Park, and A. Jain, "Soft biometric traits for continuous user authentication," Information Forensics and Security, IEEE Trans-actions on, vol. 5, no. 4, pp. 771–780, Dec 2010.

[11] T. Sim, S. Zhang, R. Janakiraman, and S. Kumar, "Continuous veri-fication using multimodal biometrics," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 4, pp. 687–700, April 2007.

[12] P. Bours, "Continuous keystroke dynamics: A different perspective towards biometric evaluation," Inf. Secur. Tech. Rep., vol. 17, no. 1-2, pp. 36–43, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1016/j. istr.2012.02.001

[13] M. Teli, J. Beveridge, P. Phillips, G. Givens, D. Bolme, and B. Draper, "Biometric zoos: Theory and experimental evidence," in Biometrics (IJCB), 2011 International Joint Conference on, Oct 2011, pp. 1–8.

[14] S. Thomopoulos, R. Viswanathan, and D. Bougoulias, "Optimal decision fusion in

multiple sensor systems," Aerospace and Electronic Systems, IEEE Transactions on, vol. AES-23, no. 5, pp. 644–653, Sept 1987.
[15] V. Nallagatla and V. Chandran, "Classifier selection using sequential error ratio criterion for multi-instance and multi-sample fusion," in Signal Processing and Communication Systems (ICSPCS), 2012 6th International Conference on, Dec 2012, pp. 1–8.
[16] K. M. Ali and M. J. Pazzani, "On the link between error correlation and error reduction in decision tree ensembles," Department of Information and Computer Science, University of California, Irvine, Tech. Rep., 1995.
[17] M. Kam, Q. Zhu, and W. Gray, "Optimal data fusion of correlated local decisions in multiple sensor detection systems," Aerospace and Electronic Systems, IEEE Transactions on, vol. 28, no. 3, pp. 916–920, Jul 1992.
[18] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," Connection Science, vol. 8, no. 3-4, p. 385403, 1996.
[19] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 405–410, Apr 1997. [Online]. Available: http://dx.doi.org/10.1109/34.588027
[20] T. Sim and R. Janakiraman, "Are digraphs good for free-text keystroke dynamics?" in IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07., June 2007, pp. 1–6.
[21] K. Chen, "Towards better making a decision in speaker verification," Pattern Recognition, vol. 36, no. 2, pp. 329–346, 2003, biometrics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/ S0031320302000341
[22] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," Automatic Control, IEEE Trans-actions on, vol. 37, no. 3, pp. 332–341, Mar 1992.
[23] D. R. Krathwohl, "A Revision of Bloom's Taxonomy: An Overview," Theory into Practice, vol. 41, no. 4, 2002.
[24] G. Doddington, W. Liggett, A. Martin, M. Przybocki, and D. Reynolds, "Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation," in International Conference on Spoken Language Processing, 1998.
[25] N. Yager and T. Dunstone, "The biometric menagerie," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 2, pp. 220–230, Feb 2010.
[26] I. Deutschmann and J. Lindholm, "Behavioral biometrics for darpa's active authentication program," in Biometrics Special Interest Group (BIOSIG), 2013 International Conference of the, Sept 2013, pp. 1–8.
[27] T. M. Cover and J. A. Thomas, Elements of Information Theory. New York: Wiley, 1 ed., 2004.
[28] Ross A, Rattani A, Tistarelli M. Exploiting the "Doddington zoo" effect in biometric fusion[C]//Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on. IEEE, 2009: 1-7.
[29] http://www.mathworks.com/help/stats/kstest2.html#btn1_1d-1
[30] Transforming animals in a cyber-behavioral biometric menagerie with Frog-Boiling Attacks; Z. Wang, A Serwadda, K Balagani, V Phoha, BTAS 2012
[31] Are Digraphs Good for Free-Text Keystroke Dynamics?, CVPR 2007, Terence Sim Rajkumar Janakiraman

[32] Bayes Error Rate Estimation using Classifier Ensembles, International Journal of Smart Engineering System, 2003, Design,Kagan Tumer, Joydeep Ghosh
[33] T. M. Cover and J. A. Thomas, Elements of Information Theory. New York: Wiley, 1 ed., 2004.
[34] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification. New York: Wiley Interscience Publication, 2 ed., 2000.
[35] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," International Journal of Man-Machine Studies, vol. 23, no. 3, pp. 263–273, 1985.
[36] J. Leggett, G. Williams, M. Usnick, and M. Longnecker, "Dynamic identity verification via keystroke characteristics," International Journal of Man-Machine Studies, no. 35, pp. 859–870, 1991.
[37] Y. Sheng, V. V. Phoha, and S. M. Rovnyak, "A parallel decision tree-based method for user authentication based on keystroke patterns," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 35, no. 4, pp. 826–833, 2005.
[38] S. M. Ross, A First Course in Probability. New Jersey: Prentice Hall, 6th ed., 2003